Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Élaboration et application de méthodes multi-critères d'analyse automatique pour 4456 parémies du Burundi

DINANT, Jean-Marc

Award date:
1985

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 04. May. 2023

85/3A/29

Facultés Universitaires Notre-Dame de la Paix

Année académique 1984-85

Elaboration et application de méthodes multi-critères d'analyse automatique pour 4456 parémies du Burundi

Par
Jean-Marc Dinant

Mémoire présenté en vue de l'obtention du grade de licencié et maître en informatique

r : Professeur Jean Fichefet

*FM
B16/1985/
26/1

BUMP

Pour mener à bien ce travail, j'ai bénéficié de la précieuse collaboration du R.P. Rodegem qui a ramené d'Afrique un corpus impressionant de parémies. Ce travail n'aurait pu être mené à son terme sans son apport et sa patience.

Jean Fichefet, professeur à l'institut d'informatique, m'a guidé vers les méthodes mathématiques ad hoc et m'a aidé à les comprendre.

Patrick Gardin, analyste-programmeur de l'A.S.B.L. Archimède, m'a fourni avec compétence et bienveillance l'environnement technique nécessaire à la réalisation de ce travail.

Anne Cornet, Myriam de St-hubert et Bruno Dinant m'ont aidé à dactylographier et à corriger cet ouvrage.

Que toutes et tous trouvent ici l'expression de ma reconnaissance.

AVANT-PROPOS

PROVERBES ET PSEUDO-PROVERBES

Diverses tentatives d'analyses informatisées d'importants corpus de proverbes se sont jusqu'ici soldées par des échecs : au Danemark (cf. Proverbium, 1969, n° 14, pp. 372-379), à Louvain (Centre de calcul, 1973), au Canada (Pierre Crépeau, 1985).

Grâce à l'amabilité et à la compétence du Professeur Jean Fichefet, l'Institut d'Informatique des Facultés Notre-Dame de la Paix a appliqué, pour la première fois, des méthodes d'analyse multi-critères à ce type de données . Les premiers résultats de cette approche semblent prometteurs. Les recherches de Jean-Marc Dinant, exposées dans le présent mémoire, vont dans cette voie.

A mon sens, les travaux antérieurs ont échoué parce que les chercheurs n'ont pas résolu les problèmes d'identification, de spécification, de classification des énoncés sentencieux qu'ils appellent "proverbes". Vu l'élasticité des termes en usage, il convient d'abord de définir les notions du genre proverbial, d'établir clairement les espèces avant d'analyser les variétés. En l'absence de définitions précises, les auteurs confondent adages, proverbes, dictons, maximes, aphorismes et autres expressions stéréotypées, extraites de leur contexte. La confusion est totale.

Dans une analyse du proverbe, ne sont pas à prendre en compte des "phrases proverbiales" ou "semi-proverbiales", ni des comparaisons telles que "vieux comme Mathusalem", "ivre comme toute la Pologne" ou "revenons à nos moutons", ni a fortiori, "de A à Z"; ce sont là des pseudo-proverbes qui ne relèvent pas de parémiologie.

La parémiologie se définit comme l'étude des énoncés sentencieux, en grec, paroimia, proverbe. Je propose de désigner d'un terme générique "parémies" l'ensemble des citations sentencieuses. Citer une parémie, c'est instaurer un échange spécifique entre un émetteur-citateur et un destinataire-cible au moyen d'un message conventionnel. Le présent résumé est une invitation à porter un regard nouveau sur ces formules typiques des cultures de l'oralité, encore bien vivaces en Afrique et qui, chez nous, sont plutôt devenues objets de curiosité pour les folkloristes.

0.1 IDENTIFICATION DU MESSAGE SENTENCIEUX

0.1.A UNE CITATION ORALE

Le mot latin pro-verbium montre bien qu'il s'agit non seulement d'une parole mise en avant, mais aussi mise à la place de... C'est toujours une citation orale mise en évidence par sa forme, son contenu et ses conditions de réalisation. la citation sentencieuse est normalement insérée dans un contexte qu'elle illustre. Ce "déjà-dit" anonyme est mis en évidence par le citateur qui instaure une rupture contextuelle, soit en recourant à des indicateurs explicites, tels que "comme on dit, comme dit le proverbe", soit en attirant discrètement l'attention par un ton de voix abaissé. En exprimant une vérité sur un ton retenu, avec la pondération qui convient au sage, il lui donne plus de poids. Cette mise en évidence est tellement discrète, qu'elle a échappé à la plupart des descripteurs.

Le citateur n'est que l'interprète de la sagesse traditionnelle. Il emprunte à la tradition mémorisée une phrase toute faite, qu'il sélectionne en fonction des circonstances. Il cite. Cité à bon escient, son message manifeste indirectement ses sentiments à l'égard de ce dont il parle : ironie, satire, cynisme, humour, selon le contexte.

0.1.B UNE FORMULE MEMORABLE

Toute citation n'est pas pour autant proverbiale. Ainsi, la formule "on n'arrête pas le progrès" est un pseudo-proverbe, car il lui manque les traits essentiels qui font le vrai proverbe. Pour avoir un proverbe, il faut que la phrase citée exprime un jugement de valeur normatif porté par les générations antérieures, coulé dans un moule rythmique, qu'il soit entériné par l'usage et tombé dans le domaine commun. Cette sentence notoire doit faire référence à une norme préétablie, édictée par l'opinion publique. Le citateur s'abrite derrière autrui. Il fait sien un argument d'autorité afin de séduire le destinataire pour mieux le convaincre.

En d'autres mots, le caractère sentencieux est le résultat d'un triple décalage : dans la forme harmonieuse, dans la structure particulière et dans la fonction du message. Tout concourt à assurer l'efficacité de la citation sentencieuse.

1. La formulation est idéalement symétrique : deux constituants sont mis en relation : "Un sou est un sou", "Santé passe richesse". La concision est de règle. Les deux constituants peuvent être qualifiés, comme dans "A bon chat, bon rat" ou "A mauvais ouvrier point de bon outil". Le message sentencieux est une unité autonome ayant un sens complet, ce qui n'est pas le cas de la comparaison "plein comme un oeuf". D'une simplicité remarquable, il est néanmoins astucieusement agencé et s'il recourt à divers artifices de langage, c'est pour attirer l'attention, s'imprimer dans les mémoires et susciter le besoin de se faire répéter. Il plaît. Proche de l'incantation scandée, il est fait pour durer, assurant du même coup la transmission et la permanence de la tradition. Dans les sociétés orales l'ensemble de ces énoncés figés assure la capitalisation du savoir.

2. L'analogie de situation. Le contenu manifeste du message est implicitement mis en parallèle avec la situation du destinataire-cible. Ainsi, pour juger un bavard, le citateur fera l'économie d'un long raisonnement en disant : "Brebis qui bêle perd sa goulée". A l'être humain, on substitue un animal, et par l'intermédiaire d'une image familière et suggestive, qui lui donne un surcroît de sens, le messager conventionnellement codé s'insinue dans l'esprit du destinataire afin d'emporter son adhésion. Le choix des figures stylistiques, qui présuppose parfois une connivence culturelle, n'est ni fortuit ni laissé au hasard.

Le message est ainsi décalé du langage courant. Trois écarts-types sont distinctifs :

1. ou bien la substitution est totale, comme dans le proverbe : "Pierre qui roule n'amasse pas mousse";

2. ou elle est partielle, comme dans l'aphorisme : "Le gibet est pour les malheureux";

3. ou elle est absente et le langage limpide est prosaïque : "Tout est bien qui finit bien".

Les énoncés sentencieux sont empiriques : ils expriment du "déjà-vu" et les rapprochements analogiques naissent spontanément de l'observation des animaux, du contact permanent avec la nature, les éléments, etc.

L' allusion à une situation fictive est fréquente, dont l'interprétation fait parfois problème. Le citateur ne risque pas de susciter l'hostilité du destinataire par une franchise trop directe. Mieux qu'un blâme direct, une fable suggestive permet de mouvoir les volontés. Ainsi, ce langage biaisé facilite la tâche de qui a des choses désagréables à dire.

3. Le contenu est latent. L'énoncé sentencieux est une forme de langage à double sens où l'essentiel n'est pas ce que l'on dit, mais ce que l'on tait, le "non-dit". Le citateur évoque sans jamais la nommer une notion abstraite. Ainsi "En close bouche n'entre mouche" conseille la discrétion, la prudence en paroles; c'est une formule normative. La norme se définit comme règle élémentaire, précepte moral, principe servant de loi.

Dans toutes les sociétés, les hommes se sont toujours imposé des règles restrictives, sans lesquelles le groupe est voué à l'anarchie. Tout groupement humain d'une certaine étendue obéit spontanément ou non à des normes de comportement, à un but pragmatique, véhiculés par les énoncés sentencieux.

0.2 FONCTIONS DE L'ENONCE SENTENCIEUX

Trois critères spécifiques sont donc nécessaires pour qu'une citation figée soit réellement une parole sentencieuse : le caractère normatif, la structure analogique et la formulation symétrique, en font un argument efficace. la formule binaire harmonieuse suggère l'équilibre souhaité par le groupe. Tout a un sens, et les leçons de choses contenues sous forme imagée ou non, dans les vraies formules sentencieuses n'ont d'autre but, en transmettant le savoir ancestral, que d'inculquer aux individus un certain conformisme social.

0.2.A DES MODELES DE COMPORTEMENT

Regroupés selon le domaine d'application, les énoncés sentencieux se subdivisent comme suit :

1. Lorsque la formule s'applique à tous les humains, la norme est dite générale. Or, la morale, l'éthique au sens large, la vie, la mort, concernent tous les hommes, partout et toujours. Il y a donc, dans un corpus d'énoncés sentencieux, une gamme d'exemples à suivre ou à rejeter qui sont d'application universelle.

2. Lorsque la formule s'applique à certains groupes d'individus, la norme est dite particulière. Sont concernées des catégories précises, comme par exemple, les acheteurs et consommateurs, ou les gastronomes. De même, la devise ne s'applique qu'à un groupe familial ou national, ou encore à un seul individu.

0.2.B BUTS DES PAREMIES.

On peut regrouper les citations sentencieuses selon leur contenu sous-jacent : d'une part les énoncés à but didactique, qui enseignent l'expérience cumulative des générations antérieures et d'autre part, les énoncés qui se présentent sous forme d'exhortation morale. Ex.: l'énoncé "Le serpent ne sera jamais ton ami" contient une norme indicative, tandis que "Hâte-toi lentement" a, de par sa norme directive, un caractère parénétique (= exhortation morale). Globalement, la parole normative évoque des obligations de modération et de prudence qui incombent à tous les hommes.

L'absence de notion normative fera écarter comme pseudo-proverbes, aussi bien la proverbiale "traînée de poudre" que des locutions telles que "une chaumière et un coeur".

0.3 CLASSIFICATION DES ENONCES SENTENCIEUX

0.3.A TYPOLOGIE DE DENOMINATION

Bien des auteurs veulent se cantonner exclusivement dans les proverbes et les dictons, alors que le genre sentencieux comporte d'autres variétés d'énoncés. Il s'agit là du problème des dénominations. De plus, après les avoir identifiées, comment classer les formules à l'intérieur d'un corpus? Le premier point concerne la typologie, tandis que le second n'est que l'application pratique des faits observés. Tous deux cependant se fondent sur le contenu sémantique.

A partir de traits distinctifs fournis par l'analyse descriptive, il est possible d'établir neuf types d'énoncés, en fonction des sortes de normes, selon la nature du contenu et des destinataires. La norme est soit générale, soit particulière.

0.3.A.1 Quelques exemples en français illustrent cette distinction. -

La norme est dite générale lorsque le contenu normatif évoqué s'applique à tous les humains.

C'est le cas pour

1. le proverbe proprement dit (P1) : "Nulle rose sans épines" comporte une norme indicative, mais il peut aussi y avoir une norme directive : "Quand le vin est tiré, il faut le boire".

2. la maxime (P2), définie comme proposition générale exprimant une norme directive, s'oppose à l'aphorisme par son contenu prescriptif : "Ne fais pas à autrui ce que tu ne voudrais pas qu'on te fit". Les modalités de la norme sont donc distinctives.

3. l'aphorisme (P3) ne peut avoir qu'une norme indicative, ce par quoi elle s'oppose à la maxime : "La crainte est le commencement de la sagesse".

4. la locution proverbiale (P4) : "Crier famine sur un tas de blé". A remarquer que (P4) contrairement aux autres parémies, peut s'intégrer dans la chaîne du discours au lieu d'en être nettement décalée, ce qui l'oppose à P1, P2 et P3. Dans les dictionnaires, (P4) est généralement reprise à l'infinitif. Exemple : "Mettre la charrue avant les boeufs". On peut donc la conjuguer, ce qui n'est pas le cas des autres parémies qui sont citées ne varietur.

La frontière entre parémie et pseudo-parémie est délicate à établir, spécialement en ce qui concerne (P4); la locution est un fait de langue qui s'insère dans le discours sans le rompre, mais qui de plus, doit évoquer une norme de comportement. On peut traduire le proverbe dans une autre langue, non les expressions idiomatiques non proverbiales, telle que "boire l'obstacle", "pêcher en eau trouble" ou "fouetter un chat".

0.3.A.2 La norme est dite particulière - lorsque le contenu normatif ne concerne que des groupes spécifiques de destinataires ou des secteurs précis d'activités.

C'est le cas pour

1. le dicton (P5), qui s'applique à diverses croyances concernant :
la prévision du temps : "Noël au balcon, Pâques aux tisons" (norme indicative), "A la Sainte-Luce, sème dru ou ne sème plus" (norme directive);
la gastronomie : "Poisson sans boisson est poison", "Qui boit meursault, meurt sot";
la caractériologie populaire : "Les yeux verts sont pervers".
Les dictons regroupent aussi des brocards visant des corporations ou des communautés locales : "Les médecins imberbes font les cimetières bossus", "Bon avocat, mauvais voisin", "Il n'y a froc si béni que le diable n'y trouve abri", "Angevin, sac à vin".

2. L'adage juridique (P6) concerne le droit coutumier : "Témoin unique est sans valeur" (plus marqué en latin : "Testis unus Testis nullus").

3. le slogan (P7) (de l'écossais sluagh ghairm: cri de guerre) est un outil de propagande sociale, commerciale, électorale. "Boire ou conduire, il faut choisir", "Au volant, la vue c'est la vie", "Pas un pas sans Bata", "Poisson de mer, santé de fer". Les slogans politiques, par nature, sont éphémères : "Giscard à la barre".

4. la devise (P8), jugement de l'émetteur porte sur lui ou sur les siens, ou injonction réflexive exprimant un idéal, ne concerne qu'une nation, une famille ou un individu : "Honni soit qui mal y pense" (Ordre de la Jarretière, G.B), "Plus oultre" (Charles-Quint).
Pour être complet, et parce qu'on les trouve dans des recueils et des dictionnaires de proverbes, il convient de ranger ici deux genres marginaux apparentés : l'apophtegme et le wellérisme.

5. l'apophtegme (P9) désigne une courte pensée, le plus souvent citation d'un auteur connu : "La critique est aisée, mais l'art est difficile" (Destouche).

6. le wellérisme (P10), plus étudié dans les pays anglo-saxons que dans les pays de langue française, est une parodie du proverbe "La vertu au milieu, comme disait le diable en se mettant entre deux prostituées".

Il faudrait encore mentionner les proverbes dialogués, qui en fait sont plutôt des fables-exprès, de brefs apologues, ou des résumés de contes. Ceci déborde du cadre des parémies.

0.3.B CLASSEMENT NOTIONNEL

Comment organiser une collection d'énoncés sentencieux? Faut-il les classer dans l'ordre alphabétique des premiers mots ou des mots thèmes, ou encore des mots clés métaphoriques? Toute classification pèche par quelque côté. Chacune a ses avantages et ses inconvénients. Le seul point de vue du lecteur doit être pris en considération et celui-ci est essentiellement pratique. Or, il est souvent difficile d'être à la fois clair, pratique et économique. Faire l'économie de redites dans les commentaires interprétatifs, dans la mention des variantes et des doublets est toujours compliqué. On peut naturellement concevoir un recueil de proverbes comme un herbier où sont 'collés' les résultats des récoltes, en faisant abstraction du milieu où vivaient - où vivent encore - ces citations orales dans tel contexte bien précis.

Pour concilier la pratique et la théorie, l'aspect utilitaire et l'aspect sémantique, il semble qu'un classement idéologique peut être à la fois simple, économique et efficace. Voici comment il est conçu : l'Homme est au centre; les besoins fondamentaux de l'Homme fournissent les grandes divisions; l'Homme est en relation

1. avec le monde qui l'entoure (Besoin de possession);

2. avec ses semblables (Besoin de socialisation);

3. avec ses supérieurs (Besoin de protection);

4. avec le monde invisible (Besoin de bonheur).

Chacune de ces quatre sections comporte des ensembles logiquement regroupés par associations d'idées. Lorsque la matière s'y prête, les catégories s'agencent dans un ordre constant : soit, par exemple, la notion de chance; en tête viendra la section 'nécessité et avantages de la chance' (le Destin). Ex. "A qui la chance sourit, tout réussit"; 'modalités d'action de la chance' (les caprices du destin). Ex. "Aux laides chattes, les beaux minous"; 'absence de chance' (la malchance). Ex. "Le gibet est pour les malheureux". Ce genre de grille permet donc de prévoir une place pour chaque énoncé et aussi, pour le lecteur, de trouver chaque formule à sa place, auprès de ses variantes éventuelles.

Le but de ce classement notionnel est de dépasser le fait concret, particulier, imagé, pour atteindre un certain niveau d'abstraction. A chaque énoncé sentencieux correspond une notion abstraite, qui autrefois lui a donné naissance. Et à chaque énoncé sentencieux, il est possible de substituer une assertion générale sémantiquement équivalente, en langage clair, directement compréhensible. Par ex. "La belle cage ne nourrit pas l'oiseau" = on peut, ayant du luxe, manquer du nécessaire.

Ainsi regroupés, les énoncés sentencieux montrent bien qu'ils traitent des problèmes communs à l'humanité. Néanmoins, lorsqu'on les compare entre eux, les proverbes du monde entier révèlent que les hommes se font du travail, de la richesse, de la femme, du pouvoir, de la mort et de l'au-delà, des idées différentes. On l'oublie parfois : les valeurs morales varient dans le te mps et l'espace. Pour cette raison, entre autres, il n'est pas inutile de tenter une analyse chiffrée du phénomène, en vue d'établir, non seulement des lois du

genre, mais aussi un profil psychologique du groupe concerné.

0.4 QUANTIFICATION DES PAREMIES

En matière d'énoncé sentencieux, que peut-on quantifier et dans quel but? Comment mettre les parémies en chiffres et pourquoi?

0.4.A LES STATISTIQUES

Il n'est certes pas facile de chiffrer les constantes observées dans les énoncés sentencieux. C'est pourtant le meilleur moyen de confirmer les hypothèses ou les intuitions. On ne trouvera ici que quelques considérations destinées à introduire les premiers résultats statistiques déjà obtenus.

Entre deux messages sentencieux, s'il y a un écart constant, on pourra en déduire qu'ils sont de nature différente. L'exemple le plus simple concerne l'écart qui existe entre la locution proverbiale et les sept autres types d'énoncés : ceux-ci sont tous stéréotypés, nettement décalés du contexte, alors que la P4 peut se conjuguer en même temps qu'elle s'intègre dans la chaîne du discours. On a donc deux ensembles qui s'opposent par une caractéristique formelle : d'une part tous les énoncés figés, d'une part, les non-figés. Tous deux peuvent faire l'objet de statistiques.

Il y a un autre mécanisme entrant en ligne de compte que l'on peut symboliser comme suit :

1. l'énoncé est totalement marqué (à tel point de vue)

2. l'énoncé est partiellement marqué.

3. l'énoncé est faiblement ou non marqué.

Ainsi par exemple, dans le cas de la substitution imagée :
"La belle cage ne nourrit pas l'oiseau", la substitution de termes est totale;
"Le gibet est pour les malheureux", la substitution est partielle;
"Témoin unique est sans valeur", le langage est clair, il n'y a pas de substitution.

Cette variation décroissante à partir d'un point maximal, jusqu'au point où le trait caractéristique est évoqué par défaut, peut être hiérarchisée grâce à l'ordinateur.

Enfin, les statistiques peuvent porter sur le degré de véracité des énoncés. Une citation n'est pas une preuve et le domaine d'exactitude est parfois contestable. Le vrai est ce qui est vérifiable. Il y a manifestement un écart révélateur entre une vérité d'évidence comme "Le soleil luit pour tout le monde" et une affirmation gratuite telle que "Qui fit Normand, il fit truand". La table de véracité comporte cinq degrés hiérarchisés comme suit :

1. vérité d'évidence : "Après la pluie, le beau temps" est une proposition qui ne demande pas de preuve;

2. proposition vraisemblable : "Femme légère a pesant mari" est un énoncé probablement vrai, dans certains cas;

3. proposition assertorique : "Hâte-toi lentement" est une proposition affirmée sans plus;

4. croyance : "Heureux au jeu, malheureux en amour", hautement subjectif, impossible à prouver;

5. affirmation gratuite : "Les médecins imberbes font les cimetières bossus".

Fondée sur la comparaison interne des énoncés, cette analyse, qui ne doit rien à l'extérieur, élimine au maximum les risques de subjectivité. Elle vise à montrer la force de persuasion du message sentencieux, son efficacité, sa dynamique propre. La fonction essentielle de ces citations orales mérite d'être mise en évidence. Ce sont, en quelque sorte, les "articles" de la jurisprudence ancestrale. On peut considérer un corpus de sentences proverbiales comme le code tacite informel d'un groupe, dont le but est de programmer les individus. Ces courtes phrases enseignent l'art de la relation humaine, elles améliorent les rapports sociaux, elles favorisent l'intégration harmonieuse de l'individu dans le groupe. L'énoncé sentencieux, on l'a dit, est vital dans les cultures de l'oralité.

0.4.B LES RESULTATS CHIFFRES

Les résultats globaux confirment la pertinence des traits distinctifs retenus : caractère normatif, structuration analogique et formulation symétrique. A noter toutefois que dans la culture orale du Burundi, il n'existe ni slogans (commerciaux ou électoraux), ni devises, ni apophtègmes. Les dictons météorologiques semblent inconnus, alors qu'en français, ils sont fort nombreux.

On trouvera ici, d'abord les chiffres globaux pour chaque type d'énoncés sentencieux, ensuite la ventilation des résultats concernant les traits distinctifs.

Dénomination des parémies	%
Norme générale	
Proverbe	63,85
Maxime	0,98
Aphorisme	1,09
Locution proverbiale	32,55
Norme particulière	
Dicton	0,20
Adage juridique	0,49
Slogan	-
Devise	-
Total	100

Formulation symétrique	%
parallélisme total	86,44
parallélisme partiel	9,10
énoncés asymétriques	4,46
Total	100

Structuration analogique	%
substitution totale	64,81
substitution partielle	24,04
énoncés en langage clair	11,15
Total	100

	%
1. L'Homme et son être	38,5
2. L'Homme et ses semblables	33,5
3. L'Homme et le pouvoir	11
4. L'Homme et l'invisible	17
Total	100

Conclusions

La parémiographie pose donc des problèmes que les informaticiens ont pris en charge. L'aspect sémantique des parémies, à lui seul, pose des difficultés qu'on espère résoudre.

En résumé, le système parémiologique comporte nécessairement trois notions essentielles : la norme (N), la structuration analogique (A) et la forme symétrique (S). On peut ici parler d'universaux parémiologiques puisque, sauf erreur, ces critères se retrouvent dans toutes les langues.

Dans l'interprétation, il arrive fréquemment qu'on trouve, même sous la plume de chercheurs chevronnés, des erreurs de décodage, qui induisent des erreurs de classement thématique. Ainsi, l'énoncé "Il vaut mieux laisser son en fant morveux que de lui arracher le nez" est dit concerner l'éducation des enfants. En réalité, le sens est le suivant : mieux vaut tolérer un mal passager que d'appliquer un remède excessif. En d'autres mots : le remède est pire que le mal. C'est la méconnaissance du principe : un proverbe vaut par sa pointe, non par ses détails, qui provoque ce genre d'erreur.

Le travail de J.-M. Dinant ouvre des perspectives intéressantes qui permettront de faire progresser non seulement la parémiologie, mais aussi la parémiométrie. Ces progrès pourraient déboucher sur des études comparatives interculturelles, soit sur la base des régions linguistiques, soit même sur des ensembles plus vastes, comme par exemple, le continent africain.

F. Rodegem

Au carrefour surprenant de la parémiologie et de l'informatique, le présent mémoire se veut avant tout un ouvrage de recherche et de réflexion. Il ne prétend pas exposer une quelconque théorie plus ou moins révolutionnaire mais espère avoir pu (dé)montrer qu'il existe dans ce domaine des possibilités réelles et prometteuses.

Il est aussi le fruit de pas mal d'erreurs et de tâtonnements, qui sont pour son auteur autant d'enseignements. La démarche suivie n'a pas été de produire un logiciel répondant à des spécifications précises, mais de tenter de répondre d'une manière générale à la question : " Peut-on appliquer des méthodes d'analyse automatique à des parémies, c-à-d à des données ayant un contenu sémantique, phonétique, syntaxique, grammatical et lexical ? ".

La démarche appliquée fut la suivante .

1. Dans un premier temps essayer de comprendre et analyser d'une manière fort intuitive le matériau qui se trouvait déjà encodé sur l'ordinateur : 4556 parémies du Burundi (+ leur traduction en français) caractérisées par cinq critères. Le codage effectué (manuellement) se prêtait peu à une analyse de données vu les énormes écarts d' effectifs entre les classes ainsi créées .

2. La création de critères supplémentaires décelables d'une manière automatique apparut rapidement comme un pas vers de nouvelles classes plus homogènes . Cela put se réaliser par une initiation(sommaire) à la syntaxe, à la grammaire et au lexique employés. Les quatre nouveaux critères retenus essaient de traduire la force homophonique de la parémie, sa structure grammaticale, le vocabulaire courant et les mots symboliques auxquels elle fait référence.

3. Enfin il fut question de choisir et d'adapter des outils mathématiques d'analyse de données, de les appliquer et d'interpréter les résultats obtenus.

Il convient ici de souligner que les programmes ont été construits en vue d'un usage rare, présentant peu d'interaction, avec un utilisateur averti. L'optique n'a donc pas été ici de fournir un produit hautement performant (vs place mémoire et temps d'exécution) mais un ensemble d'outils de recherche, capables de s'adapter aux besoins et éventuellement portables.

La machine sur laquelle ce travail a été effectué est un VAX/VMS de Digital. Il supporte les langages suivants : Pascal, Cobol, Fortran, C, Basic. Le Pascal a été choisi pour les facilités qu'il offre dans le domaine du traitement des chaînes de caractères en ce qui concerne les programmes d'établissement des valeurs des nouveaux critères . Ces programmes supposent la manipulation de fichiers pour laquelle le C est rudimentaire. En ce qui concerne les programmes plus orientés vers le calcul numérique, c'est le fortran qui a été adopté. L'interfaçage entre les deux langages se réalise via l'intermédiaire de fichiers. Cette méthode parfois un peu lente présente néanmoins l'avantage d'une portabilité parfaite. Le Fortran IV est standard. Le Pascal VAX/VMS possède des fonctions de manipulation de chaînes de caractères ne faisant pas partie de la norme Pascal standard .

Enfin, le présent ouvrage a été rédigé afin de permettre une lecture aussi compréhensible et aisée que possible, tant pour le parémiologue que pour l'informaticien. Il va sans dire que rencontrer les attentes de lecteurs aussi différents ne peut se faire sans certaines concessions. Les développements mathématiques ont été simplifiés au maximum et les explications linguistiques ne sont pas exposées dans les moindres détails.

Les annexes contiendront les formules mathématiques, programmes, et la découpe modulaire du logiciel, ainsi que certains résultats des programmes. Au début de l'annexe se trouve un plan et un mode d'emploi de celle-ci. Avant les annexes, le lecteur devrait trouver dans la bibliographie toutes les références nécessaires à l'éclaircissement de l'un ou l'autre point que l'auteur n'a pas approfondi.

CHAPITRE I

ETUDE DE L'EXISTANT

Introduction

Au début de ce travail, les matériaux suivants étaient disponibles :

1. Un fichier contenant 4456 parémies[1] du Burundi avec pour chaque parémie:

    1. Un numéro identifiant compris entre 1 et 4456.
    2. Le texte original en Kirundi.
    3. La traduction française aussi littérale que possible.
    4. Eventuellement une explication.

2. Un fichier contenant 4456 lignes de code. Chacune d'entre elles concerne une parémie et se compose de cinq nombres correspondant à une classification selon cinq critères que nous détaillerons ci-dessous.

3. Divers programmes écrits en C permettant l'édition, la sélection et l'impression de parémies possédant certaines valeurs particulières de critère. Cela autorisait des recoupements entre différents critères.

    En outre un programme offre la possibilité d'effectuer un test chi-carré[2] sur une table de contingence[1] entre deux critères distincts.

I.1 LE TEXTE EN KIRUNDI

I.1.A LES PARTICULARITES DE L'ECRITURE

Le rundi (ou kirundi) , langue dans laquelle étaient rédigées les parémies, s'écrit en utilisant les vingt-six lettres de l'alphabet ainsi qu'un certain nombre d'accents dont certains n'existent pas en

(1) Pour définition d'une table de contingence, cf. pp 22 et suiv.
(2) Pour explication du test X^2, cfr annexe E.

français. L'accentuation des lettres (uniquement des voyelles[1]) est extrêmement fréquente.

L'accentuation est présente afin d'aider à la prononciation de la langue. On peut distinguer quatre sortes différentes de tons : quatre sur les voyelles longues, quatre sur les voyelles brèves.

TABLEAU 1-1 Accentuation en Kirundi (tiré de [Rod67, pp 98])

ton	sur voyelle brève		sur voyelle longue	
bas	umugabo	homme	umutāma	viellard
haut antérieur	ukubóko	bras	umwâna	enfant
haut postérieur	umugore	femme	umwămi	roi
haut double	wyìna	danse	böse	tous

Remarque

L'absence d'accent sert à traduire un ton bas sur une voyelle brève.

Rares sont les mots en rundi qui sont différents si on tient compte de l'accentuation, mais identiques si on en fait abstraction . En outre, l'accentuation est parfois le fait de l'environnement du mot. Autrement dit, le même mot peut être accentué différemment selon les mots qui l'entourent. Pour la programmation, cela pose le problème suivant : soit on tient compte de l'accentuation des mots mais alors on arrive à déclarer identiques des mots qui ne le sont pas et vice-versa, soit on ne tient pas compte de l'accentuation et on risque aussi de déclarer identiques des mots qui ne le sont pas, mais deux mots identiques seront considérés comme égaux, malgré leur différence d'accentuation. Ces deux types d'erreurs sont bien connus des statisticiens et portent le nom d'erreur du premier et du deuxième type.

Toutefois, prendre en ligne de compte l'accentuation supposait aussi une maîtrise des subtilités de la langue dans lesquelles je n'ai pas voulu entrer. En outre, le bénéfice que l'on pourrait en retirer me semble nettement moindre que le coût de la complexité de programmation qui en découlerait.

En ce qui concerne les accents, il importe de bien distinguer trois niveaux, à savoir :

1. La manière dont ceux-ci ont été encodés.
2. La représentation interne des caractères accentués utilisés par les programmes.

(1) c-à-d "a","e","i","o","u" ; "y" est une semi-voyelle en rundi.

3. La représentation externe des caractères accentués .

La représentation de l'accentuation à ces trois niveaux est mue par trois objectifs respectifs de facilité et de rapidité de l'encodage, de facilité d'utilisation des fonctions de manipulation de chaînes de caractères (string en anglais), et enfin de qualité visuelle de la représentation sur papier. On sent bien que les conventions adoptées au premier niveau visent aussi l'objectif du niveau trois. Lors de l'encodage, une étude ultérieure du rundi n'était pas envisagée.

I.1.B LES CONVENTIONS POUR LA REPRESENTATION DE CES PARTICULARITES

I.1.B.1 Conventions d'encodage -

Les conventions d'encodage étaient imposées, puisque tout le travail d'encodage avait déjà été accompli avant de commencer ce mémoire.

Le code A.S.C.I.I.[1] alors disponible permettait de représenter en un seul caractère la plupart des caractères accentués... français . Il a donc fallu trouver des astuces pour représenter les lettres accentuées par un tilt, un accent vertical ou un accent long . Les conventions suivantes ont été adoptées afin d'avoir une représentation sur le papier propre et fidèle à l'original :

1. L'accent grave, le tilt et l'accent vertical pourront être représentés ,respectivement par les caractères ASCII "`" , "~" et "|" qui seront placés devant la lettre sur laquelle ils portent. (à et è seront encodés directement)

2. Une voyelle acentuée par un tilt ou un accent grave pourra aussi parfois être encodée en un seul caractère (è = `e ; ì = `i ; ã = ~a ; ...)

3. L'accent long, quant à lui, sera représenté par le caractère ASCII '_' qui sera placé derrière la lettre sur laquelle il porte et précédé d'un backspace[2]. Ceci a pour effet lors de l'impression de souligner d'un tiret les lettres portant l'accent long.

4. L'accent aigu pourra être représenté par un tiret bas _ (underscore) suivi du caractère sur lequel porte l'accent (sauf pour "é" qui pourra être encodé directement)

---

(1) American Standard Code for Information Inchange.
Codage associant à chaque signe d'écriture un numéro compris entre 0 et 127 ; en pratique les numéros 128 à 255 sont réservés pour des signes supplémentaires non standards, notamment pour les caractères accentués.

(2) Caractère ASCII n°8 ayant pour effet lors de l'impression d'une ligne sur une imprimante de reculer la tête d'impression d'un caractère vers la gauche. Permet donc d'imprimer deux caractères l'un sur l'autre.

Remarques

Les lettres q,l,x n'existent pas en rundi. Nous verrons plus loin [ch II, 1] pourquoi.

I.1.B.2 Conventions pour la représentation interne des caractères -

Pour la facilité de l'exploitation des fonctions de manipulation de chaînes de caractères il a semblé indispensable que la représentation interne des mots rundi jouissent de deux importantes caractéristiques :

1. Atomicité du caractère : un caractère qu'il soit ou non accentué sera représenté par un seul code ascii.

2. Unicité de la représentation : un caractère accentué ou non ne peut se traduire que par une et une seule suite de caractères A.S.C.I.I.

Suivant les conventions d'encodage, aucune de ces deux propriétés n'est acquise puisque ã , par exemple, peut s'écrire en un ou deux caractères (ã ou ~a).

Si la première propriété n'est pas respectée, la longueur d'une chaîne de caractères (c-à-d le nombre de lettres d'un mot) est rarement le nombre de caractères de celui-ci et dépend de la manière dont les lettres sont accentuées. Si la deuxième propriété n'est pas repectée, la proposition "deux mots sont identiques si et seulement si leurs représentations formelles le sont." n'est plus vérifiée.

Il va de soi que la représentation interne des caractères est dictée, pour une large part, par l'alphabet disponible sur la machine, en l'occurence un VAX/VMS.

TABLEAU 1-2 : Caractères accentués disponibles sur le VAX/VMS

Accent / Lettre	grave	aigu	circ.	tilt	trema
A	À	Á	Â	Ã	Ä
a	à	á	â	ã	ä
E	È	É	Ê		Ë
e	è	é	ê		ë
I	Ì	Í	Î		Ï
i	ì	í	î		ï
O	Ò	Ó	Ô	Õ	Ö
o	ò	ó	ô	õ	ö
U	Ù	Ú	Û		Ü
u	ù	ú	û		ü

Ce tableau peut être résumé par les règles suivantes :

1. L'accentuation possible d'une majuscule est identique à celle de la minuscule correspondante.

2. Toutes les voyelles peuvent être accentuées par un accent grave, aigu, circonflexe ou par un tréma.

3. Les lettres a,o sont les seules qui peuvent être accentuées par un tilt.

Une brève analyse des majuscules en rundi apprend que le seul accent possible sur une majuscule est un tilt. Dès lors, les majuscules dotées d'un accent grave,aigu ou circonflexe ne seront jamais utilisées. On peut s'en servir pour représenter les caractères accentués qui n'ont pas de place dans le tableau ci-dessus. La traduction entre la représentation externe lors du codage et la représentation interne se fera en adoptant les conventions suivantes :

1. Si un caractère accentué peut-être représenté par un seul caractère dans le tableau 1-2, il le sera.

2. Les lettres minuscules accentuées par un tilt seront représentées par la lettre majuscule correspondante dotée d'un accent aigu.

3. Les lettres majuscules accentuées par un tilt seront représentées par cette lettre dotée d'un accent grave.

4. Une lettre minuscule dotée d'un accent long sera représentée par la majuscule correspondante dotée d'un accent circonflexe.

Ces conventions peuvent être représentées par le tableau suivant.

Tableau 1-3 : Représentation interne des caractères rundi accentués

Accent	grave	aigu	circ.	tilt	trema	long	bref
Lettre							
A				À			
a	à	á	â	Á	ä	Â	Ä
E				È			
e	è	é	ê	É	ë	Ê	Ë
I				Ì			
i	ì	í	î	Í	ï	Î	Ï
O				Ò			
o	ò	ó	ô	Ó	ö	Ô	Ö
U				Ù			
u	ù	ú	û	Ú	ü	Û	Ü

I.2 LA CLASSIFICATION SELON CINQ MODALITES

I.2.A INTRODUCTION ET ELEMENTS DE VOCABULAIRE.

D'une manière générale, une codification selon certains critères permet d'établir des classes en vue de former une partition ou un recouvrement d'une population pouvant être représenté par une table de contingence.

Une population est une collection d'individus de la même espèce. Dans le cas qui nous occupe, la population est composée de 4556 parémies du Burundi.

Un critère est une fonction définie sur une population qui associe à chaque individu un ou plusieurs codes conventionnels[1]. En ce qui concerne le codage des parémies du Burundi, le code est toujours un nombre entier. Il sera compris entre 1 et 4456 pour le premier critère, 1 et 100 pour tous les autres critères (y compris les nouveaux). Le codage permet une classification[2] des individus de la population

C L A S S E

Une classe est un ensemble d'individus d'une population.

Elle se caractérise par

1. Un et un seul critère auquel elle appartient.
2. Une liste de codes. Dans ce qui suit nous définirons cette liste par un code minimum (ou borne inférieure) et un code maximum (ou borne supérieure). Tous les codes compris entre ces deux bornes (incluses) font partie de la liste des codes possibles de la classe.
3. L'effectif de la classe. Il s'agit du nombre d'individus (ici de parémies) qui appartiennent à cette classe.

Remarque

Le choix des codes pour un codage qualitatif est purement arbitraire. Toutefois, pour les cinq premiers critères, ce codage n'est pas aléatoire mais exprime, dans une certaine mesure, une gradation. Il n'en sera pas tenu compte dans ce qui suit. Un changement dans la numérotation des codes n'aurait donc aucune répercussion sur le résultat des analyses effectuées.

Exemple :

a) Soit P l'ensemble des étudiants de première licence en informatique à Namur

(1) cf. annexe C
(2) Nous employerons ce mot pour signifier un recouvrement (partition non-disjointe) ou une partition proprement dite

b) Soit C la moyenne de leurs cotes lors de la lère session

c) Soit A l'ensemble des élèves ayant une cote moyenne comprise entre 14 et 16

A est donc une classe d'étudiants de première licence en informatique ayant comme caractéristique commune d' avoir obtenu entre 14 et 16 de moyenne en première session.

P A R T I T I O N

Une partition est un ensemble de classes. Diviser une population en classes sur base d'un critère (le hasard peut être un critère valable) s'appelle effectuer une partition sur cette population.

Elle possède les caractéristiques suivantes :

1. Elle est disjointe : Un individu ne peut, simultanément, appartenir à plusieurs classes d'un même critère.

   Exemple : Si on considère une seule session, chaque élève ne peut appartenir qu'à une seule classe de cotations (il s'agit d'une partition). Si on considère deux sessions d'examens, un individu peut appartenir à deux classes de cotations (on parlera alors de recouvrement).

2. Elle peut être complète (ou incomplète) : Chaque individu (n') appartient (pas) à au moins une classe de la partition.

   Si on désire rendre complète une partition qui ne l'est pas, il est possible, soit de restreindre son étude à la sous-population engendrée par les individus appartenant à une classe au moins, soit de créer une nouvelle classe qui comprendra les individus non classés.

   Exemple : Si certains élèves ont été absents à certains examens, la partition est incomplète, puisque certains individus ne possédant pas de cote-code ne peuvent être classés. Si on désire rendre la partition complète, on peut, soit centrer son étude sur les élèves ayant présenté leur session en entier, soit décider de leur attribuer une cote arbitraire (0 par exemple).

3. Elle est ± homogène: Une partition est d'autant plus homogène que les effectifs des différentes classes sont proches. Il s'agit d'une caractéristique importante pour la validité de l'application de méthodes d'analyse de données. Ce point sera abordé lors du chapitre III.

   Exemple : Si tous les élèves possédaient la même cote moyenne, la partition serait on ne peut plus homogène. Si la moitié des élèves reçoivent la cote 20 et l'autre moitié la cote 0, la partition serait on ne peut moins homogène.

Une manière d'augmenter l'homogéinité d'une partition peut être de fusionner en une seule classe une ou plusieurs classes d'effectifs faibles. Une autre possibilité consiste à éclater une classe doté d'un effectif important en plusieurs classes d'effectif plus faible. Réaliser une partition relativement homogène est une des étapes préalables à l'analyse proprement dite. Elle sera plus amplement détaillée dans le chapitre IV.

La classification des parémies du Burundi selon les cinq premiers critères constitue cinq partitions complètes, en moyenne[1] fort peu homogènes.

T A B L E D E C O N T I N G E N C E

Une manière commode de présenter simultanément deux partitions de la même population est la table de contingence.

Une table T de contingence entre deux critères A et B est un tableau à N lignes et P colonnes . N est le nombre de classes de la partition engendrée par A ; P, le nombre de classes de celle engendrée par B. L'élément de la ième ligne et de la jième colonne (noté t(i,j)) est un nombre supérieur ou égal à zéro spécifiant la fréquence d'associations entre la classe i du critère A et la classe j du critère B. Cette fréquence d'association peut, par exemple, représenter le nombre d' individus (effectif) appartenant à la ième classe du critère A et à la jième classe du critère B. On parlera dans ce cas d'une table de fréquences absolues.

TABLEAU 1-4 : Table de contingence en fréquence absolue entre la dénomination et la structuration analogique de la parémie

Proverbe indicatif	336	2421	0	0	2757
Proverbe directif	22	65	0	0	87
Maxime	0	0	29	20	49
Aphorisme	0	2	999	449	1450
Locution proverbiale	18	8	13	5	44
Dicton	1	0	6	2	9
Adage	0	0	8	14	22
Apophtegme et autre	0	14	16	7	37
Total colonnes	377	2510	1071	497	4455
	Substi-tution concrète	Substi-tution abstraite	Substi-tution partielle	Parémie en langage clair	Total lignes

L'exemple ci-dessus montre la table de contingence en fréquence absolue résultant du croisement des critères deux et trois,

(1) Le lecteur pourra s'en rendre compte en considérant les effectifs des classes : seul le premier critère engendre une partition "raisonnablement" homogène. (cf. annexe C)

respectivement : la dénomination de la parémie et sa structuration analogique. Elle illustre par la même occasion le peu d'homogéinité des partitions engendrées par ces deux critères.

I.2.B CRITERE 1 : CLASSEMENT THEMATIQUE

Remarque préliminaire

> Les exemples qui suivent ont pour but de donner au lecteur un panorama des différentes caractéristiques possibles d'une parémie. Chaque exemple concerne une et une seule classe de parémies et chaque classe possède un et un seul exemple. Le lecteur désireux de saisir en détail la signification des noms de classes utilisés se réferera à l'avant-propos.

La numérotation des parémies n'est pas le fruit d'un quelconque hasard : elle se réfère à "un classement idéologique basé sur les besoins fondamentaux de l'homme" [Rod83, pp VII]. L'homme est en relation avec le monde matériel, ses semblables, ses supérieurs et l'invisible. Chacun de ces quatres thèmes peut se diviser en quatre sous-thèmes plus nuancés. Le numéro identifiant la parémie constituera le code du critère 1.

1. Le monde matériel

    1. L'avoir
       Exemple: Un grand nom n'empêche nullement une jeune fille d'être malheureuse (N°332)

    2. L'activité
       Exemple: L'oisive sarcle uniquement ses cheveux (N°770)

    3. Le savoir
       Exemple: Qui est passé en contrebas de la maison ne sait pas ce qui s'y trouve (N°1093)

    4. L'être
       Exemple: Son propre coeur pour l'individu, c'est son propre roi (N°1280)

2. L'homme et ses semblables

    1. La cohésion
       Exemple: S'embrasser est impossible si on n'a pas les bras de même longueur (N°1774)

    2. La solidarité
       Exemple: Un arbre sec s'appuie sur celui qui est vert (N°2131)

    3. L'insociabilité
       Exemple: Surveille tes paroles car la serpette est sous le lit (N°2343)

4. L'animosité
   Exemple: Il arrive que l'épieu prolonge
   la pointe de la langue (N°3015)

3. L'homme et le pouvoir

1. L'autorité
   Exemple: Lorsque le chat n'est pas là,
   les rats étalent leur queue au soleil. (N°3238)

2. La protection
   Exemple: Les vaches au milieu desquelles
   il y a un grand taureau
   ne seront pas frappées par la foudre. (N°3285)

3. La domination
   Exemple: La glycine ne discute pas
   avec la serpette (N°3523)

4. Les abus de pouvoir
   Exemple: Qui est un peu plus grand
   qu'un autre l'avale (N°3657)

4. L'homme et son destin

1. Le destin
   Exemple: Nul ne sait la direction que prendront
   en poussant les cornes d'un veau (N°3811)

2. Les aléas du destin
   Exemple: Si tu as de la malchance, la pluie
   te mouillera à deux pas d'un hameau (N°4080)

3. Le destin individuel
   Exemple: Celle qui a été esclave,
   jamais ne deviendra reine (N°4150)

4. La fatalité
   Exemple: La mort a mangé les autres,
   mais elle ne t'a pas oublié (N°4220)

I.2.C CRITERE 2 : DENOMINATION DE LA PAREMIE

Le vocable proverbe bien que fréquemment usité est inadéquat. Le proverbe n'est, en fait, qu'un cas particulier d'une parémie. On a choisi de distinguer dix types principaux de parémies :

1. Proverbe

1. Proverbe doté d'une norme indicative
   Exemple: Le désir avide de pastèque
   fait manger une courge amère (n°273)

2. Proverbe doté d'une norme directive
   Exemple: Qui va mépriser une petite parole
   regardera qui l'a prononcée (n° 1094)

2. Maxime (toujours directive)
   Exemple: Si tu as des richesses,
   tu dois les consommer (N°37)

3. Aphorisme (toujours indicatif)
   Exemple: La marche vaut mieux que l'inactivité (N°471)

4. Locution proverbiale
   Exemple: Qui passe dans un maquis inconnu
   coupera un bâton inconnu (N°1914)

5. Dicton
   Exemple: Qui a récolté un grenier de forces aura toujours
   un grenier de sorgho (N°19)

6. Adage
   Exemple: Nul ne tranche son propre procès (N°1063)

7. Apophtegme et autre
   Exemple: Les infirmités ne choissisent pas toujours
   leurs victimes (N°4241)

Remarque

La caractérisation détaillée ici est en fait incomplète. Il faudrait pour bien faire y inclure les slogans et les devises. Mais ces deux classes sont vides en ce qui concerne les proverbes du Burundi. On n'en tiendra donc pas compte.

I.2.D CRITERE 3 : STRUCTURATION ANALOGIQUE

1. Substitution totale

   1. Substitution concrète
      Exemple: Le début de la calvitie,
      ce sont les tempes dégarnies (N°3788)

   2. Substitution abstraite
      Exemple: Le début de la folie, c'est chanter (N°3789)

2. Substitution partielle
   Exemple: Si un enfant mange comme une vache,
   il mourra plus tard comme un chien (N°266)

3. Langage clair
   Exemple: Jamais un vice ne meurt,
   mais bien celui qui l'a (N°1668)

I.2.E CRITERE 4 : STRUCTURE SYMETRIQUE

1. Binaire simple v
   Exemple : Ruca-nyinshi ntíyūgárira .(N° 3691)
   La cour qui tranche beaucoup de palabres
   ne se fermera pas.

2. Binaire double |
   Exemple: Ikinyoma c'ûmwôro ntígihishwá n'urutaro. (N°3694)
   Le mensonge d'un pauvre
   n'est jamais caché par le van.

3. Tercet v
   Exemple: Umuntu w'inda ndende mu gucúra abandi acura umurambararo.
   Un gourmand pour profiter des autres
   se prétend le plus malade. (N° 246)

4. Autres procédés
   Exemple: Ijuru rivyara rídahētsé. (N° 3820)
   Le ciel engendre sans jamais avoir porté

5. Monostiche |
   Exemple : Umutûnzi ní maganya .(N°304)
   Au riche, les préoccupations.

6. Asymétrique
   Exemple: Inda yarúshije amatwí kwûmva. (N° 269)
   Le ventre entend mieux que les oreilles.

I.2.F CRITERE 5 : VERACITE.

Remarque importante

Pour des raisons indépendantes de la volonté de l'auteur, le codage de ce critère n'a été effectué que pour les 2000 premières parémies. Les conclusions concernant ce critère seront donc à prendre en considération avec une grande prudence.

1. Evidence
   Exemple: Un arbre tombe toujours
   du coté ou il penche (N°1687)

2. Vraisemblance probable
   Exemple: Un expert est toujours
   contredit par un autre (N°1383)

3. Assertorique
   Exemple: Qui a retrouvé son souffle
   Oublie ce qui l'a fait courir (N°201)

4. Croyance
   Exemple: Nul guérisseur qui exerce
   son art pour lui-même (N°1064)

5. Affirmation gratuite
   Exemple: Les Tutsi, filles de Garde-Boeuf,
   emballeront du fard, mais elles
   n'emballeront pas de provisions. (N°1308)

I.3 LA VALIDATION DES DONNEES.

Les données encodées présentaient deux propriétés qui rendaient leur validation peu commode. D'une part, le volume des données (les données représentent plus d'un demi- million de caractères) rendaient une vérification manuelle beaucoup trop lourde, d'autre part, le caractère hermétique des données en Kirundi ne permettait pas à un francophone d'en vérifier la validité, d'un point de vue orthographique par exemple.

Il convient de souligner ici l'importance de la validation des données. Les erreurs d'encodage peuvent avoir des conséquences catastrophiques. C'est ainsi par exemple que j'ai découvert, de longs mois après la mise en oeuvre des programmes, l'absence d'une parémie parmi les 4456. La parémie 2919 ne se trouve ni dans les "Paroles de Sagesse au Burundi" [Rod83] ni dans le fichier du rundi, ni dans le fichier de la traduction française. Par contre, cette parémie fantôme possède bel et bien un code selon cinq critères comme ses consoeurs. Vu que l'établissement des critères six à neuf se basait sur le rundi et/ou sur la traduction française, ceci a eu pour effet d'attribuer à la parémie 2919 le code de la parémie 2920 pour les critères six à neuf,... et ainsi en cascade jusqu'au numéro 4555. Un tiers environ de données étaient donc erronnées en ce qui concerne une comparaison entre un critère de un à cinq et un critère de six à neuf.

Au long de mon travail, j'ai eu l'occasion de déceler, le plus souvent par hasard, certaines fautes d'orthographe dans la traduction française. (shorgo au lieu de sorgho par exemple). Ceci peut avoir des conséquences néfastes sur l'établissement des valeurs des nouveaux critères. Les fautes d'orthographe sont inévitables, surtout dans un corpus aussi important et elles sont probablement plus fréquentes encore dans le texte original en Kirundi. Il existe des méthodes permettant de les diminuer[1], voire de les supprimer. J'ai supposé que les fautes d'orthographe n'étaient pas suffisamment fréquentes pour influencer "sérieusement" les résultats trouvés.

---

(1) Si on considère qu'une faute d'orthographe ne se répète pratiquement jamais et si le corpus est suffisament important pour que l' occurrence des mots soit en moyenne supérieure à un, une liste de mots apparaissant une fois seulement comprendra la plupart des fautes d'orthographe.

Par contre, l'encodage des accents a été vérifié d'une manière aussi rigoureuse que possible. Le fichier du Rundi a été examiné caractère par caractère afin de satisfaire aux tests de cohérence suivants :

1. Tous les caractères font-ils partie de l'ensemble des caractères permis ?

   Il y avait une dizaine de caractères de contrôle non-licites

2. Un backspace est-il toujours suivi d'un tiret bas ?

   Deux cas fautifs

3. Un tiret bas est-il toujours devant ou derrière une voyelle ?

   Sept erreurs

4. Un accent grave est-il toujours suivi d'une voyelle ?

   Il n'y a qu'une vingtaine d'accents graves qui sont tous suivis d'une voyelle.

Une validation de données comporte souvent certaines lacunes lorsqu'on travaille en "aveugle", c-à-d avec des données dont la taille ou l'hermétisme rendent impraticable une validation manuelle. Je pense toutefois avoir réussi à ramener le taux d'erreur à un niveau négligeable.

I.4 DEFINITION D'UN MODELE DE DONNEES ET DE FONCTIONS SUR CE MODELE.

I.4.A DEFINITION FORMELLE D'UNE PHRASE ET D'UN MOT

Trois des quatre nouveaux critères concernent la parémie originale en Kirundi. Il n'est donc pas inutile de définir d'une manière formelle la structure d'une parémie. Celle-ci sera considérée comme un standard par les modules d'établissement des critères six à huit. Il s'agit bien évidemment de la représentation interne d'un original[1] que nous appellerons par la suite phrase . Une phrase est une alternance de caractères blancs et de mots (un et un seul blanc entre deux mots) qui commence par un blanc et se termine par un blanc. Un mot est une suite de caractères licites contigus. Un caractère licite peut être:

1. Une des 26 lettres de l'alphabet (majuscule ou minuscule) sans accent.

2. Une voyelle accentuée parmi celles du tableau 1-3.

3. Une apostrophe.

(1) Parémie originale en (ki)rundi

Remarque

Il est plus pertinent de considérer l'apostrophe comme faisant partie intégrante du mot plutôt que comme un séparateur au même titre que le blanc. Celle-ci remplace en effet une voyelle disparue lors du contact avec la voyelle terminant le mot suivant. On peut consulter [Rod67, pp 110 et suiv.] pour connaître plus en détail les règles d' élision .

I.4.B FONCTIONS DEFINIES SUR CES REPRESENTATIONS

Le module "string" possède diverses fonctions ayant pour but d'uniformiser les traitements sur les originaux et sur les phrases.

En voici les principales :

Lire : Lit une parémie dans le fichier des originaux.
Transforme cette parémie en une phrase équivalente.
Cette fonction assure le passage de la représentation externe (cfr supra : conventions d'encodage) à la représentation interne (phrase).

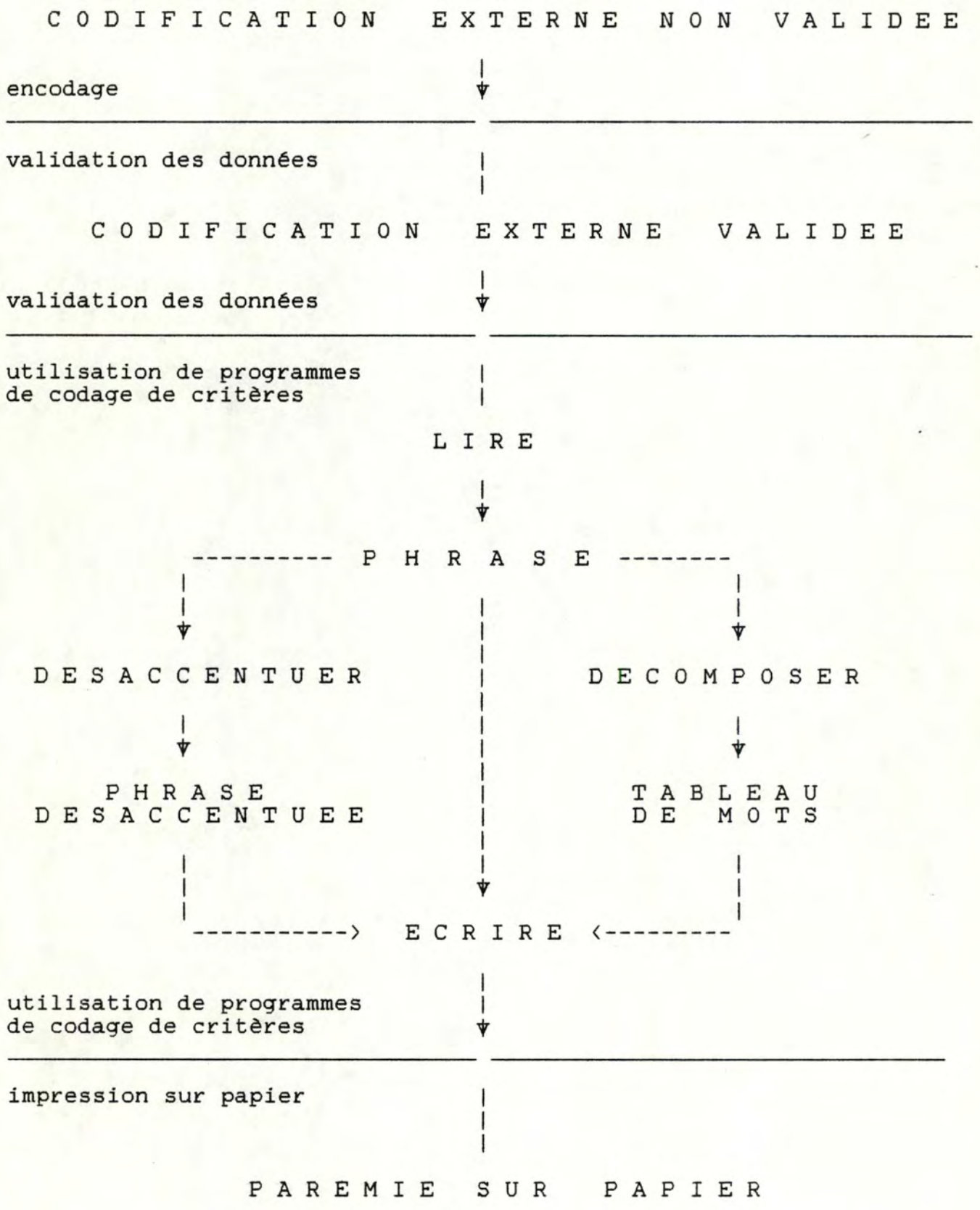
Désaccentuer : Transforme une phrase accentuée en une phrase désaccentuée.
Les mots d'une phrase désaccentuée se composent uniquement de minuscules non accentuées.

Ecrire : Ecrit une phrase accentuée ou non dans un fichier destiné à être imprimé par une imprimante.
Cette fonction assure le passage de la codification interne à la codification externe.

Décomposer : Décompose une phrase en mots.
Garantit que le mot sera un ensemble de caractères licites.

Ces fonctions sont d'une importance capitale, car elles effectuent le passage d'un énoncé peu formalisé à une représentation standard d'une parémie. Il s'agit de l'interface entre le monde réel et le modèle censé le représenter[1]. C'est la manière dont la machine "voit" les parémies. Le schéma suivant illustre le parcours possible d'une parémie et les transformations qu'elle subit.

(1) Pour une vision critique de la modélisation
Cf. conclusion du chapitre V


C O D I F I C A T I O N E X T E R N E N O N V A L I D E E
encodage
validation des données
C O D I F I C A T I O N E X T E R N E V A L I D E E
validation des données
utilisation de programmes de codage de critères
L I R E
P H R A S E
D E S A C C E N T U E R
D E C O M P O S E R
P H R A S E D E S A C C E N T U E E
T A B L E A U D E M O T S
E C R I R E
utilisation de programmes de codage de critères
impression sur papier
P A R E M I E S U R P A P I E R

CHAPITRE II

DEVELOPPEMENT DE NOUVEAUX CRITERES DE CLASSIFICATION

II.1 CRITERE 6 : CLASSEMENT PAR HOMOPHONIE

II.1.A BUT ET SENS DE CE CLASSEMENT

Le but de ce classement est de prendre en ligne de compte la force homophonique, la qualité vocale de la parémie.

La parémie étant essentiellement une coutume orale transmise de bouche à oreille, comparer des lettres entre elles pour appliquer un algorithme de détection de répétitions de sons n'a pas beaucoup de sens. Ceci pour plusieurs raisons. D'une part, deux lettres accentuées différemment ne constituent pas des sons franchement différents, mais les lettres sont formellement différentes. L'accentuation sert en effet principalement à donner une valeur musicale haute ou basse aux voyelles. D'autre part la prononciation du rundi est relativement éloignée de celle du français et cela risque fort, a priori, d'amener une multiplicité[1] de représentations formelles d'un même son.

Règles[2] principales pour la prononciation du rundi.

1. "c" se prononce "tch" comme dans "tchèque"
2. "e" est un timbre intermédiaire entre les deux sons de "étais"; il n'est jamais muet
3. "g" est toujours dur comme dans "garçon"
4. "h" est toujours expiré comme dans "hand" (main en anglais)
5. "j" se prononce "dj" souvent comme dans "djinn"
6. "n" nasalisé comme "ng" dans "sing" (chanter en anglais)
7. "o" est un son intermédiaire entre "eau" et "or"
8. "r" intermédiaire entre le "l" et le "r"
9. "s" toujours dur comme dans "son"
10. "u" toujours prononcé "ou" comme dans "cou"

(1) Dans le sens défini précédement
(2) Extrait de [Rod67, pp 12]

11. "w" comme dans "ouate". Bien qu'il soit écrit, il ne se prononce pas en rundi,
après k devant des radicaux commençant par u ou par o.
hwûbaha (respecter) se prononce kubaa
kwota (chauffer) se prononce kota

12. La voyelle terminale d'un mot ne se prononce pas, si le mot suivant commence lui-même par une voyelle.

Il fut dans un premier temps envisagé de recourir à un alphabet phonétique afin que les symboles retenus traduisent au mieux un son. Finalement, cela ne s'est pas avéré nécessaire, parce que le Rundi s'écrit d'une manière phonétique. Ceci permet d'établir la propriété d'unicité des représentations phoniques: un son s'écrit d'une et d'une seule manière. Ceci est loin d'être le cas du français. (par exemple : quand ; Caen ; cancan) . L'absence de certaines lettres dans l'écriture du Rundi [cfr note page 18] témoigne d'ailleurs de cette propriété. Le "q" est absent parce que le "k" se prononce toujours comme un "q". La règle 8 ci-dessus explique l'absence du "l". D'autre part, si le "x" était utilisé, il constituerait une redondance par rapport à "cs". En outre, l'absence des suites "tch" et "ou" dans le corpus étudié est une confirmation de ce phonétisme de l'écriture. Enfin le R.P. Rodegem, auteur de "Parole de Sagesse au burundi" [Rod83], le confirme lui-même.

Il importe de ne pas perdre de vue qu'il ne s'agit pas ici d'établir un classement quantitatif, mais un classement qualitatif permettant de regrouper des parémies présentant une similitude certaine du point de vue de leur profil homophonique. Il va sans dire que les critères retenus possèdent une part de subjectivité que leur auteur ne cherche pas à nier.

II.1.B METHODE DE CLASSIFICATION

Le codage se fera selon les critères suivants.

1. la fréquence de la répétition (simple,double,triple,...)
2. la longueur de la répétition
3. le nombre de répétitions
4. la longueur de la parémie

L'expérience, ainsi que des tatônnements successifs, nous ont poussé à tenir compte des répétitions de syllabes quelles qu'elles soient mais aussi des répétitions de voyelles séparées par des consonnes différentes (allitération). Le code du sixième critère est un nombre à deux chiffres. Le premier représente l'importance des répétitions de syllabes quelconques, le deuxième, l'importance des répétitions de voyelles. Ces chiffres seront compris entre 0 et 4

0 : homophonie insignifiante ou absente
1 : homophonie faible
2 : homophonie moyenne
3 : homophonie forte
4 : jeu de mots

II.1.C APPLICATIONS ET EXEMPLES

Afin d'éclairer ce qui précède, nous présentons ci-dessous une dizaine d'exemples illustrant chacune des classes détaillées ci-dessus:

a) v
Uramvye arabóna. (N° 900)

uramvyarabona

code répétition de syllabes : 0 (homophonie nulle)

b) v
Uramvye arambira kubóna. (N° 901)

uramvyarambirakubona
--- ---

--------››› répétition : [ram] (2 X)

code répétition de syllabes : 1 (homophonie faible)

c) Umutûnzi azira ubwôro ntázirá inzara. (N° 340)

umutunzazirubworontazirinzara
------ === ---

--------››› répétition : [nza] (2 X)
========››› répétition : [zir] (2 X)

code répétition de syllabes : 2 (homophonie moyenne)

d) Ukó birunzwé síko birungíkwa. (N° 4001)

ukobirunzwesikobirungikwa
------- -------
--------››› répétition : [kobirun] (2 X)

code répétition de syllabes : 3 (homophonie forte)

e)
```
      |
Ikibiri kibí kirîyiririza. (N° 655)

 ikibirikibikiriyiririza
 ===-+++==-==+++ +++++

--------)))  répétition  : [ikibi] ( 2 X )
========)))  répétition  : [iki] ( 3 X )
++++++++)))  répétition  : [iri] ( 4 X )
```

code répétition de syllabes : 4 (jeu de mot)

f) Ndarinze yagûye mu murinzi. (N° 1423)

```
          aieaueuuii
```

code répétition de voyelles : 0 (homophonie nulle)

g)
```
                  v            v
Nyen'imbúgitá n'uyifashe ikirindi. (N° 3659)

eiuiauiaiiii
  -- --

--------)))  Répétition  : [ui] ( 2 X )
```

code répétition de voyelles : 1 (homophonie faible)

h) Igikûndiro c'ûmwâna kiva kurí nyina. (N° 2587)

```
iiuiouaaiauiia
--==      ==-

--------)))  Répétition partielle : [ii] ( 2 X )
========)))  Répétition partielle : [ui] ( 2 X )
```

code répétition de voyelles : 2 (homophonie moyenne)

i)
```
  |     v
Uburo bwinshi ntíbugirá umusurúru. (N° 658)

          uuoiiiuiuuuuu
          --      -----
--------)))  répétition  : [uu] ( 4 X )
```

code répétition de voyelles : 3 (homophonie forte)

j)
```
          _           v
Amanyáma yoshwa n'agashambara. (N° 1313)

          aaaaoaaaaaa
          ---- -----
--------)))  répétition  : [aa] ( 7 X )
```

code répétition de voyelles : 4 (jeu de mot)

II.2 CRITERE 7 : CLASSEMENT PAR STRUCTURE GRAMMATICALE

Le but de ce classement est de prendre en considération la structure grammaticale de la parémie originale. La classification se fait d'une manière fort simple en se basant sur la présence de certains indices dans la parémie. Ces indices sont soit un mot, soit un affixe, soit encore une série d'affixes.

Les différents types de structure grammaticale retenus sont

1. Proposition nominale.

   Exemple : Uhimvye ntâ kindi aganyá. (indice : "ntâ")

   Qui est rassasié ne se plaint
   d'aucune autre chose. (N° 1)

2. Proposition nomino-verbale.

   Exemple : Amângati aruta isari. (indice : "(-)ari")

   Renvois d'estomac (satiété)
   valent mieux que fringale. (N° 6)

3. Proposition verbale

   Exemple : Amatûnga aragōra Ndíbutungé yībagira Ndíbutumbé.
   (indice : "Nd" + 4 lettres au moins)

   Avoir du bien fait problème : (qui dit:)
   "J'ai la possibilité de devenir riche" oublie
   "Je crèverai peut-être avant". (N° 42)

4. Proposition subordonnée

   Exemple : Ukubá mu bitōke síko kumōta ibigomvyi.
   (indice : "sí" + 3 lettres au plus)

   Etre au milieu des bananiers
   ce n'est pas nécessairement sentir
   les feuilles de bananiers. (N° 100)

5. Proposition subordonnée sans conjonction

   Exemple : Hākó uzimíra wozígura. (indice : "Hāko")

   Au lieu de t'égarer,
   mieux vaut que tu fasses un détour. (N° 748)

6. Proposition subordonnée conjonctionnelle

   Exemple : Ntāwuvûka rimwé ngw arāré akúze. (indice : "ngw")

   Nul être en naissant, ne peut
   prétendre être adulte en une nuit. (N° 860)

7. Proposition avec verbe auxiliaire
   v

   Exemple : Umugabo ntámerá ubwanwa aba ahéjeje.
   (indice : "ba" à la fin d'un mot
   de cinq lettres au plus)

   Un adulte n'a de la barbe que
   lorsqu'il est pleinement mature. (N° 991)

Un des grands problèmes rencontrés lors de cette classification est le nombre élevé de parémies non classées. Il n'a pas été possible de ramener le nombre de propositions indéfinies à un chiffre inférieur à deux milles. Cela est certainement dû à un manque de connaissances de la langue d'une part, et d'autre part, à la rudimentarité de l'algorithme employé.

Durant la mise au point de ce critère, un va-et-vient continuel s'est déroulé entre le linguiste et l'analyste-programmeur. Une des conclusions à en tirer est qu'il est fort difficile pour un non-informaticien de donner des critères objectifs voir rigides permettant d'établir d'une manière sûre et constante la structure grammaticale. L'analyste possède certes l'habitude de créer ce type de règles d'une manière efficace, mais la connaissance de la langue lui fait cruellement défaut.

Ici aussi, il s'agit donc d'une classification correcte mais incomplète

II.3 CRITERE 8 : CLASSEMENT PAR MOTS CLES

II.3.A BUT ET SENS DE CE CLASSEMENT

Le but de ce classement est sémantique et consiste à établir un critère similaire au premier critère d'une manière plus différente, moins subtile, mais automatique.

II.3.B LA METHODE ET LE PROGRAMME DE CLASSIFICATION

L'idée d'effectuer un classement à partir de mots clés peut paraître à priori passionnante, surtout d'un point de vue sémantique. D'emblée, on se heurte à des problèmes importants parmi les suivants :

1. Comment choisir les différents mots clés afin que ceux-ci puissent être "révélateurs"
2. Quel nombre de mots clés choisir ?
3. Quelle tactique adopter si une parémie comporte plusieurs mots clés ?
4. Faut-il choisir les mots clés dans la langue d'origine ou dans la traduction ?

5. Si on les choisit dans la langue d'origine, comment déceler un même sème sous des contenus lexicaux différents ?

J'ai choisi dans un premier temps de considérer les mots les plus fréquents ayant une valeur sémantique, à savoir les substantifs, adjectifs . Le rundi est riche en particules de toutes sortes n'ayant aucune valeur sémantique précise ou constante. Les verbes ne seront pas pris en ligne de compte, non parce que leur contenu sémantique est insignifiant mais parce que la complexité de leurs règles de formation rend trop difficile leur analyse. Il y a en effet plus d'un millier de formes verbales possibles pour un même thème, mais seulement "quelques" unes sont utilisées dans le langage courant. Il n'existe pas de critère non sémantique et non contextuel (lié à la représentation formelle du mot par exemple) qui permette de distinguer les formes permises de celles qui ne le sont pas. Pour prendre une comparaison dans la langue française : rien dans le signifié d'un verbe, n'indique s'il est transitif ou non.

Les méthodes d'analyse de données choisies (voir chapitres suivants) supposent d'avoir des classes relativement homogènes. Dans le cas contraire, les résultats obtenus sont peu parlants. Dans le choix des mots clés, seuls ont été retenus les mots clés se repétant un nombre de fois supérieur à 10 sur l'ensemble des 4555 parémies.

En outre, quelque soit le choix des mots clés retenus et pour peu qu'il y en ait un nombre raisonnable, il peut arriver que plusieurs mots clés se trouvent dans une même parémie. Or, il n'est pas possible, a priori[1], qu'une parémie possède plus d'un code pour un critère donné. Un choix s'impose.

La tactique qui a été appliquée consiste, dans un premier temps, à construire une liste d 'occurrence de tous les mots clés. Lors de l'analyse proprement-dite, si une parémie contient plusieurs mots clés, c'est le moins[2] courant qui sera retenu. Ce choix peut paraître surprenant mais il présente l'énorme avantage de réduire l'effectif des classes de mots clés courants et d'augmenter celui des classes de mots clés plus rares et donc d'avoir une distribution d'effectifs plus constante

La langue originale fera l'objet de notre étude. La traduction risque d'être trop riche et ainsi de multiplier le nombre de mots clés possibles. Il faudrait ensuite regrouper sous un même thème les mots clés ayant des significations sémantiquement connexes, ce qui ne pourrait se faire sans l'intervention d'un être humain, puisque les critères de regroupement sont sémantiques.

Ainsi par exemple le thème"vache", fondamental dans la vie et la culture du burundi, pourrait être composé des mots français suivants : bétail, bovin, veau, genisse, taureau, vache,... Tous ces mots pourront se trouver au singulier ou au pluriel, au masculin ou au féminin.

Le rundi contrairement à la langue française peut nuancer un même thème de nombreuses manières différentes . Lors de l'introduction

(1) Nous verrons dans le chapitre III qu'il est possible de supprimer cette restriction en créant une recouvrement.
(2) L'utilisateur du programme peut décider le contraire.

d'un mot clé, on veillera à donner une indication sommaire sur la nature de celui-ci.

Un mot pourra ainsi être

1. Une structure nominale (i.e. un substantif ou un adjectif) : il peut alors se décliner de 19 façons différentes.

2. Un verbe : il peut alors se conjuguer d'une manière extrêmement complexe.

3. Un invariant : il ne peut changer, et c'est littéralement le mot clé communiqué qui doit se retrouver dans la parémie en rundi. Ceci permet de prendre en considération des mots clés grammaticalement irréguliers.

Notons que dans les deux premier cas, l'utilisateur communiquera uniquement les thèmes des mots. Le programme développé permet en outre de regrouper plusieurs mots clés dans une même classe, il suffit pour cela qu'ils aient même traduction.

Le problème de découvrir un même mot clé apparaissant sous des formes différentes sera résolu en appliquant les règles suivantes :

1. Si un mot pris littéralement est identique à un mot clé invariable, alors on retient ce mot clé.

2. Si un mot clé est identique à un thème de mot clé décliné (s'il s'agit d'une forme nominale) ou conjugué (s'il s'agit d'un verbe), alors on retient ce mot clé. Cette analyse se déroulera en deux temps.
   Dans un premier temps, le mot en rundi sera comparé à tous les thèmes des mots clés suivis/précedés des combinaisons de préfixes/suffixes ad hoc.
   Si cette méthode ne donne aucun résultat, on tiendra compte des contractions/élisions et autres phénomènes qui peuvent se produire au contact des différentes parties du mot.

3. Si, après l' analyse de tous les mots de la parémie, aucun ne correspond à un mot clé, on regarde, dans la traduction de la parémie, si se trouve la traduction d'un mot clé. Si le résultat est positif, alors on retient le mot clé.

4. On examine alors la liste des mots clés retenus, si elle est vide, on donne à la parémie un code spécial signifiant "sans mot clé"; sinon on lui donne le code du mot clé le moins courant (cfr supra).

II.3.C DEVELOPPEMENT D'UN ANALYSEUR SYNTAXIQUE

II.3.C.1 Aperçu de la construction des mots en rundi -

II.3.C.1.a Introduction -

La syntaxe rundi est extrêmement riche en préfixes et en suffixes. Si le nombre et la diversité de ceux-ci peut paraître, a priori, un obstacle à une analyse syntaxique, on se rend compte, a posteriori, que ce foisonnement d'affixes de toutes sortes rend l'analyse plus complexe, mais aussi possible. En langue française , par exemple, il n'y a souvent rien dans le signifiant d'un mot qui objectivement permette de le classer dans une catégorie syntaxique déterminée. Tout mot ayant un contenu sémantique raisonnable (verbe,substantif, adjectif) répond toujours à la structure suivante :

```
=============================================
! * préfixe(s)  ! un thème ! * suffixe(s) !
=============================================
```

* signifie Zéro,un ou plusieurs

II.3.C.1.b Les formes nominales -

Ce sont les mots qui ont la structure la plus simple puiqu'ils se composent uniquement d'un préfixe classificateur et d'un thème. Parfois, le thème peut être redoublé, il ne sera pas tenu compte de ce phénomène rare dans le cadre de cet analyseur syntaxique. Notons aussi qu'un adjectif peut être en général substantivé en ajoutant un augment. L'augment sera toujours une voyelle identique à celle qui termine le préfixe qui le suit.

Exemples[1]

1. mu=kuru= : ancien,supérieur et u-mu=kuru= (l'aïeul)
2. u-mu=ntu= : un homme
3. a-ba=ntu= : des êtres humains
4. i-ki=ntu= : être matériel, chose
5. ru=ntu= : humainement
6. a-ka=ntu= : petite chose
7. u-tu=ntu= : petites choses
8. a-ha-=tu= : un lieu
9. u-ku=ntu= : manière d'être

(1) tirés de [Rod67, p.14]

II.3.C.1.c Les formes verbales. -

Les verbes, quant à eux, se forment d'une manière extrêmement complexe puisqu'ils peuvent comporter cinq préfixes et trois suffixes. [Rod67, tableau p.42]

II.3.C.2 Définition d'un modèle de données. -

Dans la définition de ce modèle de données, trois objectifs ont été poursuivis :

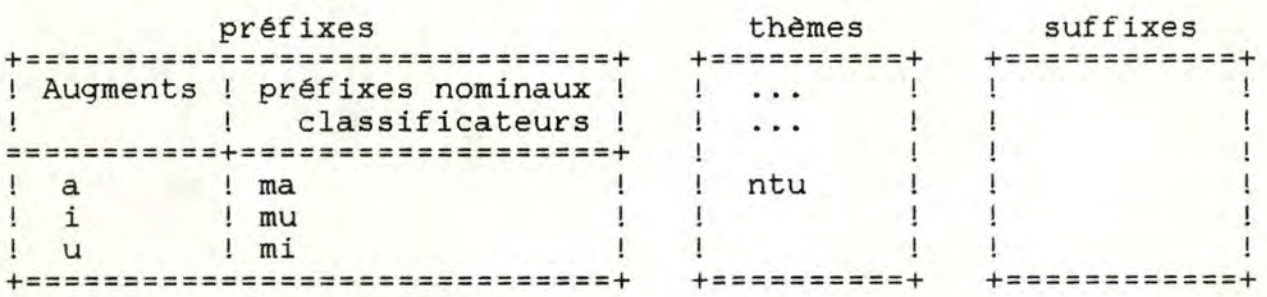
1. Procéder d'une manière incrémentale, c-à-d partir d'une définition élémentaire en lui ajoutant, au fur et à mesure de la rencontre des obstacles, des raffinements successifs pour en venir à bout.

2. Opter résolument pour la simplicité de la structure des données, même s'il s'ensuit que les algorithmes les manipulant sont peu performants.

3. Ne pas vouloir faire entrer à tout prix prendre en ligne de compte les phénomènes marginaux.

Si on choisit de représenter les différentes parties du mot (pts[1]) comme les différents noeuds d'un graphe, la structure syntaxique d'un mot peut se représenter comme un chemin dans ce graphe.

Exemple : le chemin traduit la structure syntaxique de u-mu=ntu=.



Cet exemple simple suppose que :

1. Le parcours de noeud en noeud est unique, c-à-d qu'il n'existe qu'une seule décomposition pts cohérente. Ceci n'est pas toujours vrai : en ce qui concerne les formes verbales par exemple, les thèmes verbaux peuvent être précédés de cinq types de préfixes différents : préinitiale, initiale, postinitiale, marque, infixe (respectivement); or il se fait que l'affixe "ha" peut être soit une initiale, soit un infixe . Il est important de savoir si "ha" est l'un ou l'autre, car dans le premier cas il peut être suivi d'une marque,alors que dans le second cas ce n'est pas possible.

(1) Préfixe - Thème - Suffixe

Dans le cadre de cette étude, il n'est pas intéressant de pouvoir analyser le temps ou le mode du verbe ; il suffit simplement de pouvoir retrouver son thème. Ce problème sera résolu dans la construction de l'analyseur.

2. Il n'existe pas d'affixe répétitif. En pratique, en ce qui concerne les formes verbales, ceci est souvent faux. Dès lors, il nous a paru bon de distinguer dans un premier temps deux formes d'affixes : simple ou répétitif. L'expérience nous a poussé à nuancer cette division en ajoutant une troisième forme : la forme combinatoire . l'affixe combinatoire est un affixe répétitif qui a comme particularité que la même forme de l'affixe ne peut se répéter. Un affixe répétitif peut se voir comme une série d'affixes simples. Nous laisserons de côté dans le cadre de cette analyse les affixes combinatoires en les considérant comme des affixes répétitifs.

3. Les affixes sont tous obligatoires : si c'est vrai dans le cas des substantifs et des adjectifs, il n'en est pas de même pour les formes verbales. Le problème sera résolu en appliquant la règle suivante : si un affixe est facultatif on ajoutera aux valeurs possibles de cet affixe la valeur "" (le mot vide) que nous noterons par la suite Ø.

CONCLUSION

Une structure syntaxique peut être définie comme une suite ordonnée d'affixes ou encore comme le parcours d'arcs dans un graphe dont les sommets représentent des affixes. L'affixe est une partie du mot qui peut prendre un nombre fini de valeurs, dont éventuellement la valeur Ø . Un même affixe peut être répété un nombre fini de fois.

II.3.C.3 construction de l'analyseur -

Un "bon" analyseur devrait être complet et correct. Complet en ce sens qu'il doit toujours produire une réponse face à n'importe quelle proposition. Correct parce qu'il ne peut évidemment pas se tromper. Mon ambition a été de développer un prototype correct mais incomplet. Construire un analyseur syntaxique complet est pratiquement impossible sans prendre en considération le contexte. En effet, en français comme en rundi, deux mots ayant formellement le même signifiant (c-à-d qui s'écrivent en utilisant exactement les mêmes signes dans le même ordre) peuvent appartenir à des structures lexicales différentes. Pensez par exemple au mot "parent" en français.

En outre, les capacités de l'analyseur seront de déterminer si le mot qui lui est soumis est un verbe, une forme nominale ... ou autre chose, et de transmettre le thème du mot s'il a réussi à le classifier dans une des trois premières catégories. La version actuelle de l'analyseur possède une table des thèmes et invariants possibles.

Dans le cas de l'analyse d'une forme verbale[1], comme vu supra, il se peut qu'un même mot puisse obéir à plusieurs décompositions pts différentes. Dans ce cas, si le thème trouvé n'est pas le même pour toutes les décompositions, l'analyseur classera le mot parmi les indéterminés.

En ce qui concerne l'introduction des affixes, l'utilisateur devra entrer pour chaque affixe les renseignements suivants :

1. Le nom de l'affixe (préfixe nominal,initiale,augment,...)

2. Les valeurs possibles de l'affixe (par exemple pour un adjectif, les préfixes peuvent être "mu", "ba", "mi", "ji", "ki", "bi", "ru", "tu", "bu"). Si l'affixe est facultatif, il aura soin d'ajouter dans la liste le mot vide. Si l'affixe peut se répéter n fois, il le répètera n fois.

3. Pour chaque structure, l'utilisateur devra indiquer le nombre de préfixes et le nombre de suffixes.

Remarques

Il est facile de traduire "l'affixe X doit se répéter au moins m fois et peut se répéter au plus p fois" . Il suffit de donner l'affixe X n+m fois ; n fois sans le mot vide dans la liste de ses valeurs possibles ; m fois avec le mot vide dans la liste de ses valeurs possibles.

L'ordre dans lequel seront entrés les affixes a une importance : c'est l'ordre dans lequel ils devront trouver place dans le mot.

Un problème de taille est celui de la contraction ou de l'élision de la voyelle terminale du préfixe suite au contact de celle-ci avec un thème commençant par une voyelle. Ce problème a été résolu par l'usage de grammaires terminales. Le principe en est simple. On possède une liste d'affixes et leurs traductions. La liste d'affixes est en fait la résultante d'une contraction d'affixes et la traduction est la concaténation des affixes originaux. La traduction d'un affixe peut être récurrente en théorie, mais dans le cas présent, ceci est superflu. Afin d'éclairer ce qui précède, voici un bref exemple d'une grammaire transformationnelle et une de ses applications.

traduction original

ubwa => ubua
ubwe => ubue
ubwu => ubuu

La flèche "=>" peut se traduire par "vient de ". Prenons l'exemple du mot "roi" qui a comme thème "ami". Lors de la déclinaison avec le préfixe "ubu", le "u" devant le "a" devient un "w", ce qui donne "ubwami".

(1) Cette partie de l'analyseur est encore au stade du projet.

La résolution du problème de détection des contractions réside dans l'application chronologique des règles suivantes jusqu'à un échec, ou jusqu'à la découverte d'un thème.

1. Si le mot commence par un préfixe connu, ce mot dénudé de son préfixe sera le thème.

2. Si le mot commence par un préfixe inconnu, consulter les grammaires terminales afin de trouver une concaténation d'affixes (original) constituant un préfixe présent au début du mot considéré.

3. Si une telle concaténation n'existe pas, alors on aboutit à un échec.

4. Si elle existe, alors remplacer le préfixe original par sa traduction et retourner avec le mot ainsi modifié à l'étape un.

L'implémentation des grammaires transformationnelles [Cho69] peut se réaliser de deux manières. La première que je qualifierai de méthode active consiste à coder les règles de contractions dans un algorithme qui est exécuté en temps réel, lors de l'analyse d'un mot. La deuxième méthode est passive et suppose l'intervention d'un opérateur humain. Celui-ci aura pour tâche d'énumérer, non pas des règles, mais la liste exhaustive des contractions possibles.

La faiblesse principale de cette méthode est l'importance du travail nécéssaire afin d'établir une liste exhaustive de toutes les transformations possibles. Le cas particulier de la contraction des préfixes nominaux classificateurs est un exemple rare d'application possible de la méthode passive. Pour les formes verbales, par exemple, une telle liste me semble peu réalisable. Par contre, cette solution présente un avantage évident de souplesse et diminue le risque d'erreur.

La souplesse est augmentée parce que l'apparition d'une nouvelle règle ou d'une variante d'une ancienne règle ne nécessite pas la recompilation de l'algorithme. La grammaire transformationnelle se trouve en effet sur fichier et peut être modifiée à tout moment grâce à un éditeur de texte. La modification d'une règle voit ainsi son effet directement répercuté dans le comportement de l'analyseur syntaxique.

Le risque d'erreur est minimisé. En effet l'erreur provient le plus souvent d'une règle défectueuse (certaines "exceptions" n'ont pas été prises en ligne de compte). Enoncer une règle signifie rarement que son auteur aie en tête tous les cas possibles et imaginables auquels elle puisse s'appliquer. Il se trouve constamment dans un processus de va-et-vient entre l'établissement de variantes et exceptions et un processus d'expérimentation de celles-ci.

Ce va-et-vient fut un des handicaps pour l'établissement des codes de la structure grammaticale. Le linguiste énonce des règles que la machine applique d'une manière on ne peut plus systématique. Mais les résultats produits ne rejoignent que rarement ses attentes à cause de tel ou tel cas particulier... d'où nouvelles règles ... et nouvelles imperfections.

Ecrire une liste exhaustive de toutes les contractions possibles, résultantes de la rencontre entre un préfixe et un thème, revient certes aussi à écrire une règle, mais dans un champ d'application nettement restreint.

Enfin, la méthode détaillée ci-dessus possède plusieurs faiblesses dont la plus importante est l'ignorance de la multiplicité possible des originaux possibles pour une même traduction. Ceci peut être clairement illustré par l'exemple du thème "ana" (enfant). Précédé du préfixe "aba", le "a" débutant le thème et le "a" final du préfixe, se contractent en un "a" long et donnent donc le mot "abana". Lors de la décomposition du mot, l'analyseur, trouvant le préfixe "aba" au début du mot, en déduit que le thème du mot est "na", ce qui est faux. Un mot comme "abana" est une forme ambique

Le lecteur saisira ici toute la pertinence des "invariants" . Si le nombre de thèmes est faible, les formes ambigues peuvent être encodées comme invariantes et renvoyer au mot concerné. Ainsi le thème "ana" sera décomposé en l'invariant "abana" et le thème déclinable "ana". Ceci suppose évidemment une intervention manuelle.

II.4 CRITERE 9 : MOTS SYMBOLIQUES

Dans la traduction française, certains mots sont marqués d'une astérisque. Il s'agit de mots à prendre dans un sens symbolique détaillé à la fin de "Paroles de Sagesse au Burundi"[Rod83]. On compte 501 astérisques qui renvoient à 68 mots formellement différents.

Tableau 2-1 : Occurrence des mots-symboliques

CHIEN	59	ROI	54	LAIT	22	BIERE	18
PLUIE	18	NUIT	16	MARMITE	15	TAMBOUR	13
GRENIER	13	FAMINE	13	GALE	12	DENT	12
PROVERBE	11	YEUX	11	COEUR	9	BELLE-MERE	9
JAMBE	9	HYENE	9	ELEPHANT	8	SORGHO	8
RAT	8	VEAU	7	TAUREAU	7	HUTU	7
VENT	7	FOUDRE	7	CHEVRE	7	PYGMEE	7
MAQUIS	6	VAN	6	FOURMI	5	SUIE	5
SEL	5	TESTICULE	5	JOUR	5	SERPENT	5
LANCE	5	TUTSI	4	MOU	4	LEOPARD	4
ROSEE	4	BARATTE	4	PORTE	3	SPATULE	3
BANANIER	3	MARTEAU	3	BATON	3	VENTRE	3
TABAC	3	BEURRE	3	BOUCLIER	3	GENDRE	2
PRECIPICE	2	COURGE	2	EPAR	2	OEIL	1
TAURILLON	1	MIEL	1	PATE	1	MEULETON	1
ORPHELIN	1	GENIE	1	BANANE	1	SHORGO	1
GENISSE	1	MEULE	1				

Le faible volume des données à considérer permet une vérification manuelle de l'établissement de ce critère. Ainsi, vu qu'il y a 501 astérisques dans le fichier du français et que le nombre de mots symboliques est lui-aussi de 501, aucun mot-clé ne peut avoir échappé au programme.

Une brève analyse des mots ci-dessus a permis de faire, a priori, certains recoupements. Par exemple, "bananier" et "banane" seront regroupés sous le même vocable "banane", "oeil" sera confondus avec "yeux", "shorgo[1]" avec "sorgho", etc... .

Pour des raisons techniques, le nombre de classes par critère était limité à cinquante pour la première méthode d'analyse de données appliquée. J'ai appliqué la méthode de hiérarchisation afin de regouper certaines classes d'effectifs faibles en classes d'effectif plus important. Cette méthode se trouve détaillée au chapitre III. Certains mots symboliques de fréquence 1 ont été éliminés car ils paraissaient vraiment trop peu significatifs.

Lors de la construction de la table de contingence, une classe "Sans symbole" contenant les parémies inclassables (i.e. sans aucun mot-symbolique) fut créée. Elle a été supprimé par la suite. Le critère 9 concerne donc seulement 501 parémies. Il s'agit ici de la restriction de l'étude à une sous-population[2] de parémies.

(1) Notons que la méthode proposée au chapitre I [note p.27] serait efficace pour corriger cette faute d'orthographe.
(2) cf. p. 22

CHAPITRE III

CHOIX DE METHODES D'ANALYSE DE DONNEES

Préliminaires

Dans la présentation de ce qui suit, l'auteur a été soucieux d'assurer au lecteur l' intelligibilité la meilleure possible. Le développement mathématique a cédé la place à un discours moins formel et plus intuitif, qui , nous l'espérons, sera à la portée du parémiologue. Les méthodes appliquées ne sont pas neuves et ont fait l'objet d'une bibliographie abondante à laquelle le lecteur sera fréquemment renvoyé.

La démarche adoptée dans le choix de ces méthodes ne sera pas détaillée ici. Elle a été guidée par trois critères principaux qui sont la fiabilité (choisir des méthodes bien documentées, ayant déjà fait leurs preuves et dont les résultats soient vérifiables), la compréhensibilité (les résultats qui peuvent se traduire par un graphique sont plus facilement compréhensibles par un néophyte) et enfin la faisabilité (qui suppose la maîtrise par l'auteur des outils mathématiques ad hoc, la prise en compte de la complexité de la programmation et la disponibilité éventuelle de logiciels mathématiques fiables).

Bien évidemment, seules les méthodes d'analyse qualitative ont été prises en ligne de compte. Le but poursuivi par ces méthodes est de parvenir à éprouver la cohérence des critères de classification et de découvrir s'il existe des rapports entre ces critères. Les chapitres suivants se chargeront de l'interprétation de la nature des rapports entre les différents critères.

Enfin, il convient de souligner que les 4456 parémies du Burundi sont censées constituer une liste quasi-exhaustive de toutes les parémies en cours dans ce pays.

III.1 LE PROBLEME DES VALEURS MULTIPLES POUR UN SEUL CRITERE.

Un problème majeur rencontré lors de la mise en oeuvre de nouveaux critères de classification peut être résumé par la question suivante : " Quelle valeur de critère attribuer à un individu pouvant simultanément être classifié par plusieurs valeurs de ce critère ? ".

Si on se réfère strictement à la définition, il s'agit bel et bien d'une fréquence d'associations entre deux groupes de valeurs de critères. Dès lors, rien ne s'oppose à ce qu'un individu puisse posséder plusieurs valeurs d'un même critère, si ce n'est l'augmentation de son poids dans l'analyse de données.

Imaginons par exemple qu'une étude souhaite mettre en évidence les relations existant entre la marque de cigarette fumée par les conducteurs de voiture habitant Namur et la marque de la voiture qu'ils possèdent. Il est dans ce cas évident qu'un namurois peut posséder zéro, une ou plusieurs voitures et fumer zéro,une ou plusieurs marques des cigarettes. Pour simplifier cet exemple, ne considérons que deux marques de voiture (Renault et FIAT) et deux marques de cigarettes (Belga et Gauloise).

Les valeurs possibles pour le critère de la marque de voiture sont les suivantes.

Un namurois peut

1. ne posséder aucune voiture
2. posséder uniquement une Fiat
3. posséder uniquement une Renault
4. posséder une Fiat et une Renault

Il peut également

1. ne pas fumer
2. fumer uniquement des "Belga"
3. fumer uniquement des "Gauloise"
4. fumer chacune de ces deux marques

Le but de l'étude étant de découvrir les liens entre la marque que fume le namurois, et la voiture dans laquelle il roule, on peut très bien concevoir que le namurois qui roule en fiat et en Renault et, qui fume des "Gauloise" et des "Belga", engendre une augmentation d'effectif dans quatre intersections de classes (Fiat-Gauloise ; Fiat-Belga ; Renault-Gauloise ; Renault-Belga). On appelera multiplicité d'un critère le nombre de classes distinctes de ce critère auxquelles un même individu peut appartenir.

Remarquons que la multiplicité d'un critère dépend donc de la partition choisie. Dans l'exemple, accepter une multiplicité de deux pour le critère de la marque de cigarettes suppose implicitement que l'on accorde autant d'importance à un namurois qui fume deux marques de cigarettes, qu'à deux namurois qui fument une seule marque de cigarettes. Cette remarque peut constituer un critère d'acceptation ou de refus d'une multiplicité supérieure à un.

L'établissement de tous les critères a été bâti de manière à ce que l'utilisateur puisse choisir la multiplicité voulue d'un critère donné. Ayant comme objectif une analyse de données probante, j'ai personnellement accepté une multiplicité supérieure à un lorsque la distribution des classes y gagnait en homogenéité.

III.2 RECAPITULATION ET EXTENSION DU VOCABULAIRE

Nous nous trouvons en présence d'une population d'individus (en l' occurence de parémies) caractérisés par une ou plusieurs valeurs de neuf critères appelé(s) code(s). Ces codes permettent de répartir la population entière en un certain nombre de classes. Pour deux critères donnés, l'intersection de leurs classes se résume par une table de contingence en fréquence absolue donnant, pour chaque couple de classes de critères différents, le nombre de parémies appartenant simultanément à la première classe et à la deuxième.

III.2.A REPRESENTATION GEOMETRIQUE D'UNE CLASSE : LE NUAGE DE POINTS.

Reprenons l'exemple vu de la page 22. Chaque dénomination possède une distribution dans l'ensemble des structurations analogiques possibles. Braquons notre attention sur les lignes de ce tableau. Faire une comparaison entre deux dénominations du point de vue de leur structuration analogique revient à comparer deux lignes de ce tableau. Pour rendre cette comparaison plus facile, on préfère souvent construire la table de contingence en fréquences relatives. Il s'agit de la table en fréquence absolue ramenée en pourcentage du nombre d'individus de la population observée.

Ce tableau rend nettement plus aisée une comparaison des colonnes ou des lignes, simplement par une diminution de grandeur des chiffres employés. Nous verrons lors de l'analyse des correspondances qu'il existe d'autres tables plus "parlantes".

Choisissons de regrouper en une seule classe les susbtitutions concrètes et abstraites afin de former la classe des substitutions totales. On obtient la table suivante

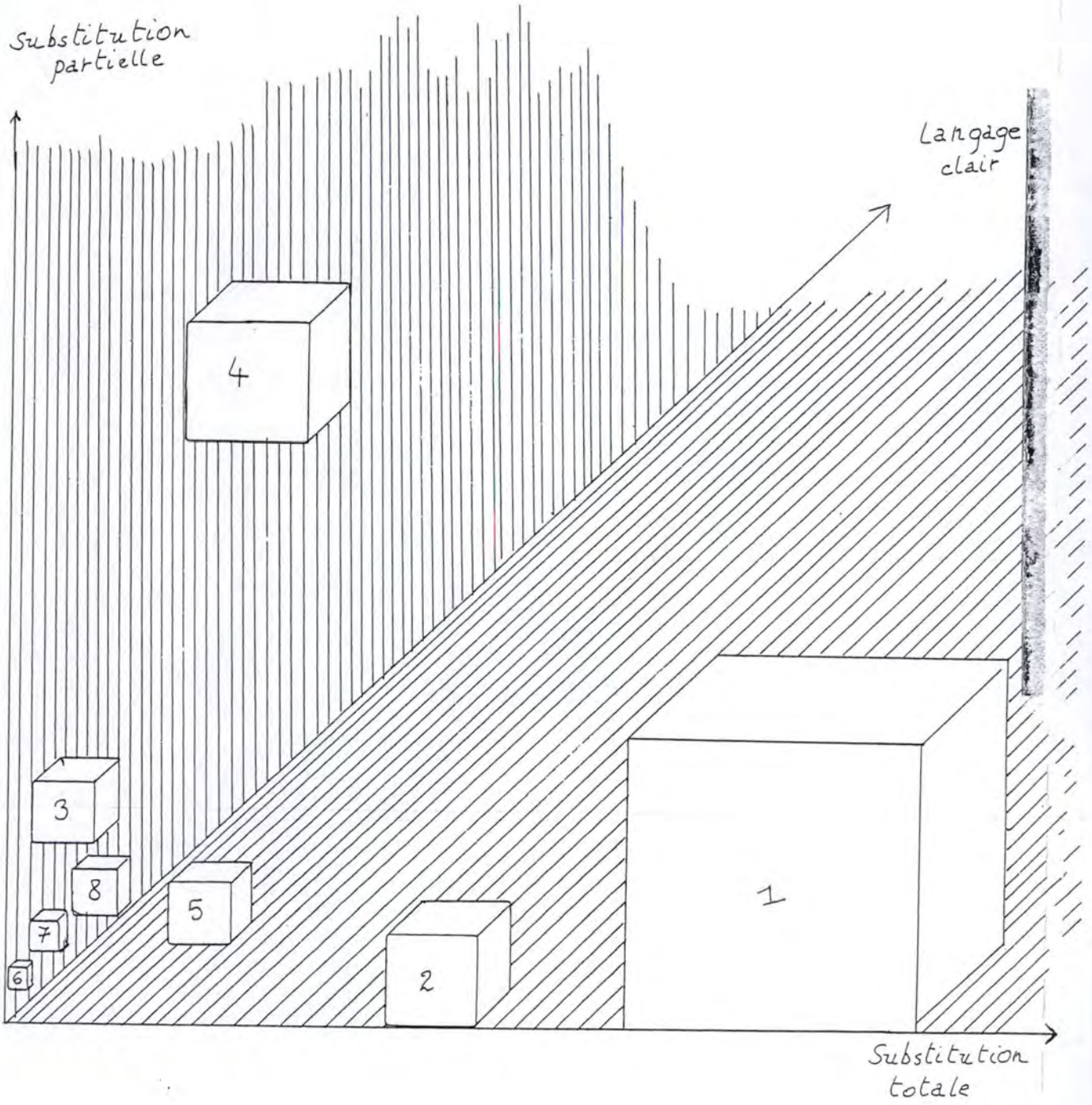

Substitution partielle
Langage clair
4
3
8
5
7
6
2
1
Substitution totale

Tableau 3-1 : Dénomination et Analogie en valeur absolue

1 Proverbe indicatif	61.9	0.0	0.0	61.9
2 Proverbe directif	2.0	0.0	0.0	2.0
3 Maxime	0.0	0.7	0.4	1.1
4 Aphorisme	0.0	22.4	10.1	32.5
5 Locution proverbiale	0.6	0.3	0.1	1.0
6 Dicton	0.1	0.1	0.0	0.2
7 Adage	0.0	0.2	0.3	0.5
8 Apophtegme et autre	0.3	0.4	0.2	0.8
Total colonnes	64.8	24.0	11.2	100.0
	Substi-tution totale	Substi-tution partielle	Parémie en langage clair	Total lignes

Nous pouvons représenter les huit types de dénomination dans l'espace à trois dimensions de la structuration analogique.

Du graphique en regard de cette page, ressort nettement la structuration analogique de la parémie selon sa dénomination. On constate, par exemple, que les proverbes, qu'ils soient indicatifs ou directifs penchent nettement vers la substitution totale. La taille du cube symbolise le poids de la classe qu'il représente (ici le pourcentage de parémies comprises dans cette classe).

Tous ces cubes constituent donc un nuage de points dotés d'un poids. Une autre caractéristique importante de ce nuage est son centre de gravité. Physiquement, le centre de gravité d'un objet est un point de cet objet tel que, si on plantait cet objet sur une aiguille, il resterait parfaitement en équilibre. Il importe de constater que le centre de gravité d'un nuage de points n'est pas nécessairement un point de ce nuage. En pratique, c'est même rarement le cas.

L'inertie d'un nuage est la distance moyenne de ses points par rapport au centre de gravité. Il s'agit en quelque sorte de la densité du nuage. Si les points du nuages sont très clairsemés, l'inertie du nuage sera faible. Si le nuage est très compact, avec des points très proches les uns des autres, son inertie sera élevée.

Cette représentation d'une classification par un nuage de points est très parlante mais elle impose une limitation importante.

Si un critère possède plus de 3 classes, la représentation des classes d'un deuxième critère en fonction des classes de ce critère devient impossible. En effet, une représentation géométrique dans une espace de plus de trois dimensions n'est pas visualisable. Les critères retenus possèdent parfois plus d'une cinquantaine de classes, Il faut donc trouver une représentation autre mais parlante, capable de représenter une partition de ce type. On parlera dans ce cas de méthodes d'analyse multi-critères

III.3 L'ANALYSE DES CORRESPONDANCES

Préliminaires

Afin de mieux faire comprendre au lecteur ce dont il s'agit, le développement théorique sera accompagné d'un exemple mené de bout en bout. Cet exemple est le croisement entre la dénomination et la structuration analogique de la parémie (critère 2 et 3). Tous les résultats de calculs proviennent du logiciel que j'ai utilisé ou créé.

La méthode d'analyse des correspondances s'applique à des tables de contingence quelconques. Elle "permet de décrire les proximités existant entre les profils-colonnes et profils-lignes, en tenant cependant compte des différences d'effectifs entre ces lignes et ces colonnes" [Leb79, p 309].

Considérons la table de contingence suivante :

tableau 3-2 : Dénomination et structuration analogique (fréquence absolue)

Proverbe indicatif	7.5	54.3	0.0	0.0	61.9
Proverbe directif	0.5	1.5	0.0	0.0	2.0
Maxime	0.0	0.0	0.7	0.4	1.1
Aphorisme	0.0	0.0	22.4	10.1	32.5
Locution proverbiale	0.4	0.2	0.3	0.1	1.0
Dicton	0.0	0.0	0.1	0.0	0.2
Adage	0.0	0.0	0.2	0.3	0.5
Apophtegme et autre	0.0	0.3	0.4	0.2	0.8
Total colonnes	8.5	56.3	24.0	11.2	100.0
	Substi-tution concrète	Substi-tution abstraite	Substi-tution partielle	Parémie en langage clair	Total lignes

La fréquence de l'intersection de deux classes de deux critères distincts est rarement révélatrice. Par exemple, savoir que 10.1 % de parémies sont des locutions proverbiales en langage clair est d'un intérêt tout relatif. Ce à quoi on aimerait parvenir serait de pouvoir trouver des "classes qui se ressemblent". Intuitivement, on sent bien que deux classes d'un critère donné qui se ressemblent devraient avoir des distributions très semblables dans les classes de l'autre critère. Ainsi, par exemple, si nous prenons la colonne 3 et 4 du tableau 3-2, nous constatons -toujours intuitivement- qu'elles ont à peu près le même "profil". C'est le cas aussi pour les colonnes 1 et 2. Par contre, les couples de colonnes (1,3) (1,4) (2,3) (2,4) ne se ressemblent vraiment pas.

On peut de la même manière, considérer les lignes du tableau 3-2.

Encore faut-il parvenir à définir une distance entre deux lignes ou colonnes. Cette notion est fondamentale car c'est elle qui permet de quantifier la ressemblance entre deux lignes ou deux colonnes. Nous ne nous étendrons pas ici sur ce problème de la distance et renvoyons le lecteur désireux d'en savoir d'avantage à l'annexe E.

La distance utilisée pour l'analyse des correspondances est celle du X^2. Cette distance possède l'avantage de vérifier le principe d'équivalence distributionnelle[1] : éclater une classe en plusieurs ou rassembler différentes classes en une seule ne modifie pas la distance des autres classes entre elles.

III.3.A REPRESENTATION GRAPHIQUE

A ce stade-ci, remarquons que la perception que nous avons d'un nuage de points est totalement dépendante de la forme du nuage, certes, mais bien aussi de l'endroit d'où est vu le nuage. Ceci est fondamental, car c'est de l'angle sous lequel le spectateur regardera le nuage qui conditionnera la compréhension du relief du nuage.

Prenons l'exemple du vélo. Si un martien[2] débarquant sur la terre apercevait un vélo de face, il ne verrait qu'une sorte de ligne verticale du genre.

Cette vision du vélo n'est guère éclairante. Par contre, la vision du vélo vu de profil le renseignera sur ce qu'est un vélo : il verra alors apparaitre un pédalier, des roues, des rayons, etc... Un des buts de l'analyse des correspondances est de présenter le nuage de points de manière à rendre compte au maximum de la structure du nuage.

Il est possible de calculer, pour chacun des deux nuages de points, des axes d'inertie maximum passant par le centre de gravité du nuage. On appelle ces axes "axes principaux". L'axe d'inertie maximum est une droite passant dans le nuage et qui se rapproche le plus possible d'un nombre maximum de points. Ainsi, l'axe d'inertie maximum d'un vélo, est une droite qui ,approximativement , passe par le moyeu de la roue arrière et par celui de la roue avant.

On assure ensuite une rotation du nuage de points de manière à présenter de profil l'axe d'inertie maximum du nuage. Ceci n'est pas suffisant. En effet imaginons un vélo horizontalement couché par terre tel que son axe d'inertie maximum (la droite passant par les deux moyeux) soit perpendiculaire à notre regard, un martien ne verrait qu'une ligne horizontale dénué de signification.

La solution est de présenter le vélo de manière à ce que son deuxième axe d'inertie maximum (il s'agit d'une droite passant approximativement par le moyeu du pédalier et par la selle) soit lui aussi perpendiculaire au regard. Nous obtenons alors un vélo de profil et debout, ce qui permet, même à un martien, de comprendre ce qu'est un vélo.

---

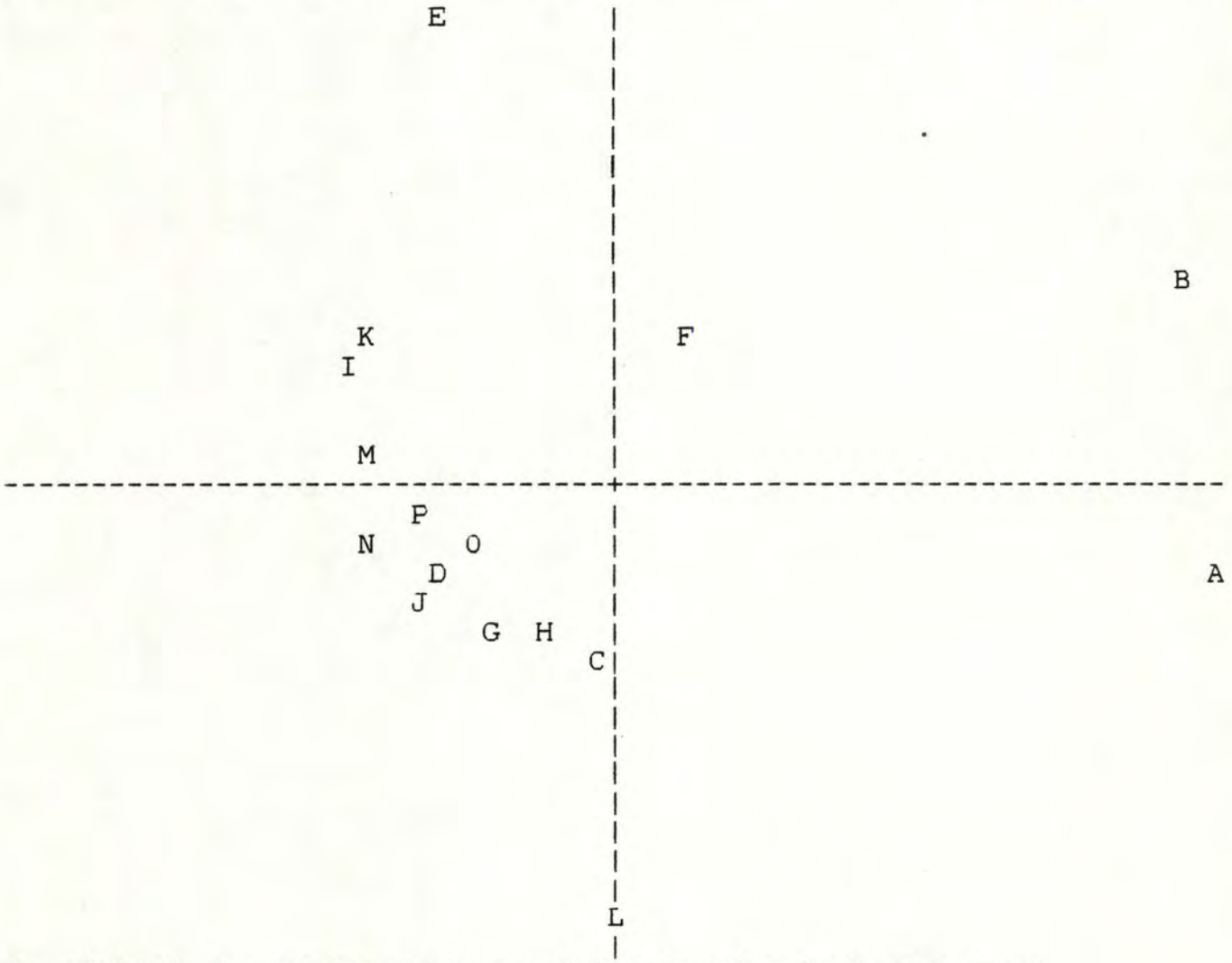
(1) Cf. Annexe E.

(1) L'analyseur de parémies ressemble un peu à un martien : il doit interpréter des choses qui lui sont, a priori, étrangères.

III.3.B INTERPRETATION GRAPHIQUE

On tient généralement compte de la projection suivant les deux axes d'inertie les plus importants. Il est possible de calculer le pourcentage d'inertie expliqué par la représentation graphique. Il est possible de représenter simultanément les points lignes et les points colonnes sur un même graphique. Cette possibilité sera exploitée lorsqu'il ne risque pas d'y avoir un excès de points dans le graphique.

Exemple : projection du thème suivant la dénomination de la parémie



En réalité, le graphique est : 2.2 fois plus haut que large

A:Avoir	462	B:Activité	385	C:Savoir	420
D:Etre	451	E:Cohésion	409	F:Solidarité	200
G:Insociabilité	490	H:Animosité	389	I:Autorité	65
J:Protection	215	K:Domination	148	L:Abus de pouvoir	77
M:Destin	124	N:Aléas du destin	296	O:Destin individue	79
P:Fatalité	245				

Inertie expliquée : 59.84% + 23.95% = 83.79% de l'inertie totale

La proximité entre deux points de même nature (i.e. deux points-lignes ou deux points-colonnes) indique une corrélation entre les deux classes qu'ils représentent. Elle signifie que la distribution en pourcentage des deux points présente un profil semblable. Cependant, la proximité des points peut être faussée par la déformation due à leur projection. La deuxième méthode (hiérarchisation) palliera à cette ambiguité.

Un point proche de l'origine signifie qu'il possède une distribution proche de la distribution moyenne des autres points du nuage.

En considérant la position des points sur les axes principaux, il est en général possible de déduire le sens de l'axe.

L' analyse sera d'autant plus probante que le pourcentage d'inertie du nuage expliqué par les axes principaux retenus sera élevé.

La distance entre deux points de nuages différents signifie qu'en moyenne l'effectif de l'intersection entre les deux valeurs de critère qu'ils représentent est élevé.

III.3.C LE PROGRAMME

Le programme a été conçu d'une manière interactive. Il est capable de bâtir le graphique de l'analyse en composantes principales pour le croisement de n'importe quels critères. Il est possible de projeter les points-lignes et/ou les points-colonnes. En outre, lors d'une session, l'utilisateur peut supprimer les classes qui lui semblent trop isolées des autres classes afin d'obtenir une représentation plus parlante. Dans ce cas, l'analyse ne porte plus évidemment que sur une partie de la population. L'usage le plus répandu en analyse des correspondances est de représenter simultanément,sur un même graphique, les projections des points-lignes et des points-colonnes. Cette possibilité ne sera pas envisagée ici, car cette manière de présenter les choses alourdit parfois considérablement le graphique et n'aide guère le linguiste à la compréhension.

Le graphique apparaît à l'écran suivant la hauteur et la largeur désirées. Les points sont aussi éloignés que possible du centre, mais aucun d'eux ne se perd hors du graphique. L'échelle de projection horizontale est différente de l'échelle de projection verticale (toujours pour avoir le graphique le plus "aéré" possible). Après une exécution du programme, les graphiques successifs se trouvent dans un fichier que l'utilisateur peut faire imprimer sur papier.

Le calcul des valeurs propres est réalisé par le logiciel Eispack© [Eis78] écrit en fortran et transformé pour gérer la double précision. L'appel aux sous-routines Fortran se fait à partir d'une procédure Pascal sans paramètres. Afin d'assurer un portabilité raisonnable, tous les paramètres sont passés par fichier (les mécanismes de passage de para mètres entre langages différents sont en effet fort dépendants de l'environnement hardware et software).

A la fin de l'exécution du programme, l'utilisateur peut sauver la nouvelle partition qu'il vient d'effectuer. Le sauvetage modifie uniquement les classes des critères concernés par l'analyse. Ceci permet d'utiliser d'autres méthodes d'analyse sur une partition plus pertinente.

Remarque concernant le temps d'exécution

Le calcul des valeurs propres, bien que réalisé par un logiciel performant, accapare le plus gros du temps calcul de ce programme. La taille de la table de contingence est limitée à 50 et le temps d'exécution est proportionnel au carré de la taille de la table. Il faut compter plusieurs minutes de temps C.P.U. dès que la taille approche 40.

III.4 LA CLASSIFICATION ASCENDANTE HIERARCHIQUE

Préliminaires

L'analyse des correspondances présente une faiblesse : elle montre la proximité des points, mais laisse à l'utilisateur le soin de regrouper les points entre eux selon sa subjectivité. En outre, la distance entre points est sujette à caution parce qu'elle peut être biaisée par la mauvaise qualité de la représentation.

Il existe des moyens mathématiques aptes à chiffrer la qualité de la représentation d'une part, la corrélation entre des points-variables d'autre part. Un des buts du logiciel créé est de produire des résultats parlants, surtout pour le parémiologue. Les tableaux de chiffres ne possèdent, pas selon moi, cette propriété de compréhensibilité par tout un chacun.

La méthode de hiérarchisation ascendante développée ci-après offre l'avantage d'une compréhension quasi-immédiate en évitant de devoir recourir à des tableaux de chiffres

III.4.A APPROCHE DE LA METHODE ET ELEMENTS DE VOCABULAIRE.

Une hiérarchie[1] est un ensemble de classes possèdant certaines caractéristiques fondamentales.

1. Elle est complète[1] : l'union des classes de la hiérarchie vaut la population entière et chaque individu appartient à au moins une classe de la hiérarchie.

2. deux classes de H sont incluses l'une dans l'autre ou alors leur intersection est vide.

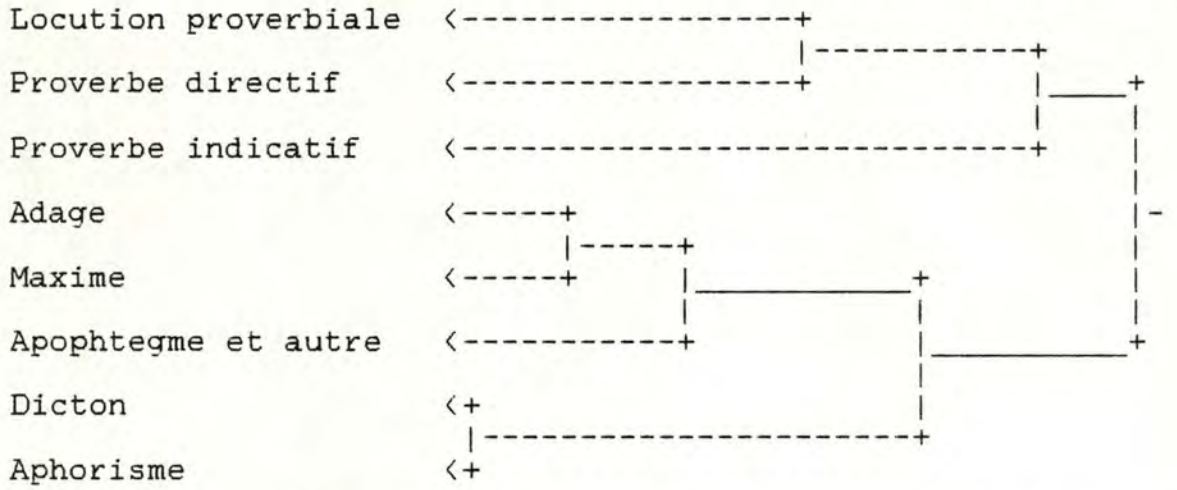
Il y a deux manières de réaliser une hiérarchie. La première méthode consiste à diviser la population en deux classes de manière à maximiser la distance entre les deux classes. Ensuite on s'attaque séparément à chaque classe et on recommence jusqu'à , par exemple, qu'il y aie au plus N individus par classe. On appelle cette méthode "descendante" parce qu'elle construit l'arbre à partir du haut[3].

La méthode ascendante consiste à essayer d'agréger les deux classes les plus proches l'une de l'autre de manière à former une seule classe... et ainsi de suite jusqu'au moment où toutes les classes sont réunies en une seule classe : la population toute entière.

III.4.B LE GRAPHIQUE

Quelque soit la manière dont on construit une hiérarchie, elle peut se représenter à l'aide d'une structure d'arbre.

Exemple : Hiérarchie de la dénomination selon l'analogie.



La corrélation entre deux variables sera d'autant plus forte que le noeud reliant ces 2 variables est à gauche du graphique.

---

(1) Pour une définition plus formelle, voir [Bou81, pp 41 et suiv]
(2) Bertier [Bou81, pp 41] emploie le mot "totale".
(3) En informatique, les branches des arbres sont tournées vers le bas

III.4.C LE PROGRAMME

Le programme est celui qui se trouve dans "Traitements de données statistiques" [Leb79, pp 407 et suiv.]. Il a été modifié au niveau des entrées/sorties afin d'assurer une présentation un peu moins sommaire des résultats, et une largeur du graphique qui puisse tenir sur une feuille din/A4. Il permet de bâtir une hiérarchie ascendante indicée en utilisant au choix une des quatres distances suivantes : saut minimal, écart moyen, variance et barycentre. L'expérience montre que la variance permet en moyenne le graphiques les plus probants.

III.5 COMPLEMENTARITE DES DEUX METHODES.

Les résultats de la hiérarchisation complètent d'une manière avantageuse ceux fournis par l'analyse des correspondances. Graphiquement, on peut schématiser sur le graphique des correspondances une hiérarchisation ascendante en appliquant les règles suivantes:

1. Entourer d'un cercle les deux classes les plus proches en se basant sur les résultats de la hiérarchisation.
2. Considérer chaque cercle comme une classe.
3. S'il existe encore des classes non encerclées, aller à 1, sinon stopper.

On appliquera cette méthode dans le chapitre suivant.

CHAPITRE IV

APPLICATION DES METHODES D'ANALYSE DE DONNEES

IV.1 INTRODUCTION

Nous avons donc élaboré une classification des parémies selon neuf critères. Grâce à cette classification nous pouvons bâtir des tables de contingence entre tout couple de critères i,j (i,j = 1,...,9). Nous possédons deux outils d'analyse de données que nous pouvons appliquer à ces tables de contingence. Il serait possible de se lancer dans une étude systématique de ((9 x 9) - 9) / 2 soit 36 tables de contingences[1].

Cependant ne serait-il pas plus économique d'entreprendre d'abord, comme les statisticiens en ont l' habitude, une sorte de "pré-étude"? Celle-ci pourrait réveler si, a priori, il n'y a quasiment pas de chance pour que deux critères donnés soient dépendants . Ceci fut effectué par le biais de test chi-carrés. Les résultats étaient absolument fantaisistes[2] et ne donnaient guère d'indication sur la dépendance ou non des classes analysées. Nous avons donc envisagé un examen table par table.

Le lecteur trouvera ci-après quatre résultats choisis parmi les plus probants, ainsi qu'une interprétation. D'autres résultats qui peuvent mériter une quelconque attention se trouvent dans l'annexe D. Mon but est de montrer par ces exemples qu'une analyse informatique des parémies ne relève pas de l'utopie propre à certaines recherches en "intelligence" artificielle et d'ainsi esquisser une réponse empirique à la question soulevée lors de l'introduction.

Dans la présentation des résultats qui suivent, seuls les résultats probants seront interprétés. Il va de soi que mon étude ne se limite pas aux seuls graphiques présents dans cet ouvrage. Certains croisements ne faisant pas ici l'objet d'une interprétation se trouvent en annexe.

---

(1) Un croisement entre un critère et lui-même n'a aucun sens ; Un croisement entre un critère A et un critère B est identique à un croisement entre B et A

(2) Pour détails, cf. annexe E.

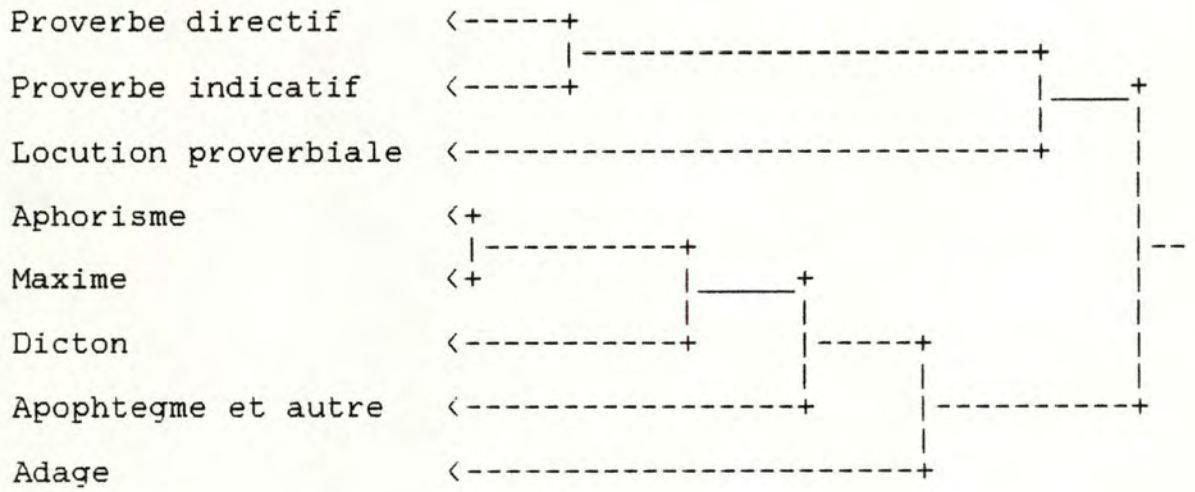
IV.2 LIEN ENTRE ANALOGIE ET DENOMINATION

Table de contingence en valeurs absolues

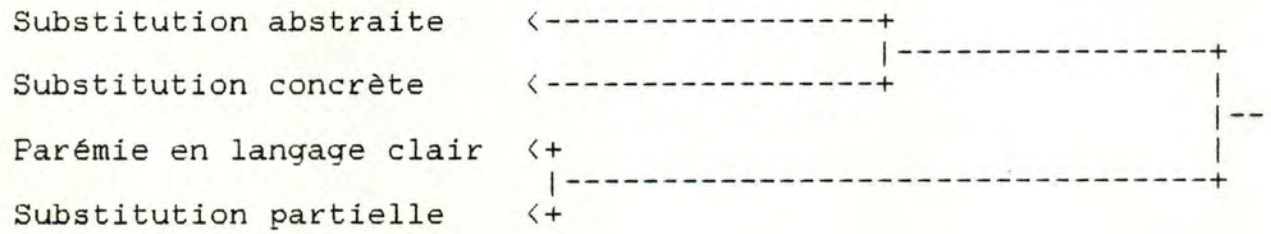
Proverbe indicatif	336	2421	0	0	2757
Proverbe directif	22	65	0	0	87
Maxime	0	0	29	20	49
Aphorisme	0	2	999	449	1450
Locution proverbiale	18	8	13	5	44
Dicton	1	0	6	2	9
Adage	0	0	8	14	22
Apophtegme et autre	0	14	16	7	37
Total colonnes	377	2510	1071	497	4455

Substitution concrète
Substitution abstraite
Substitution partielle
Parémie en langage clair
Total lignes

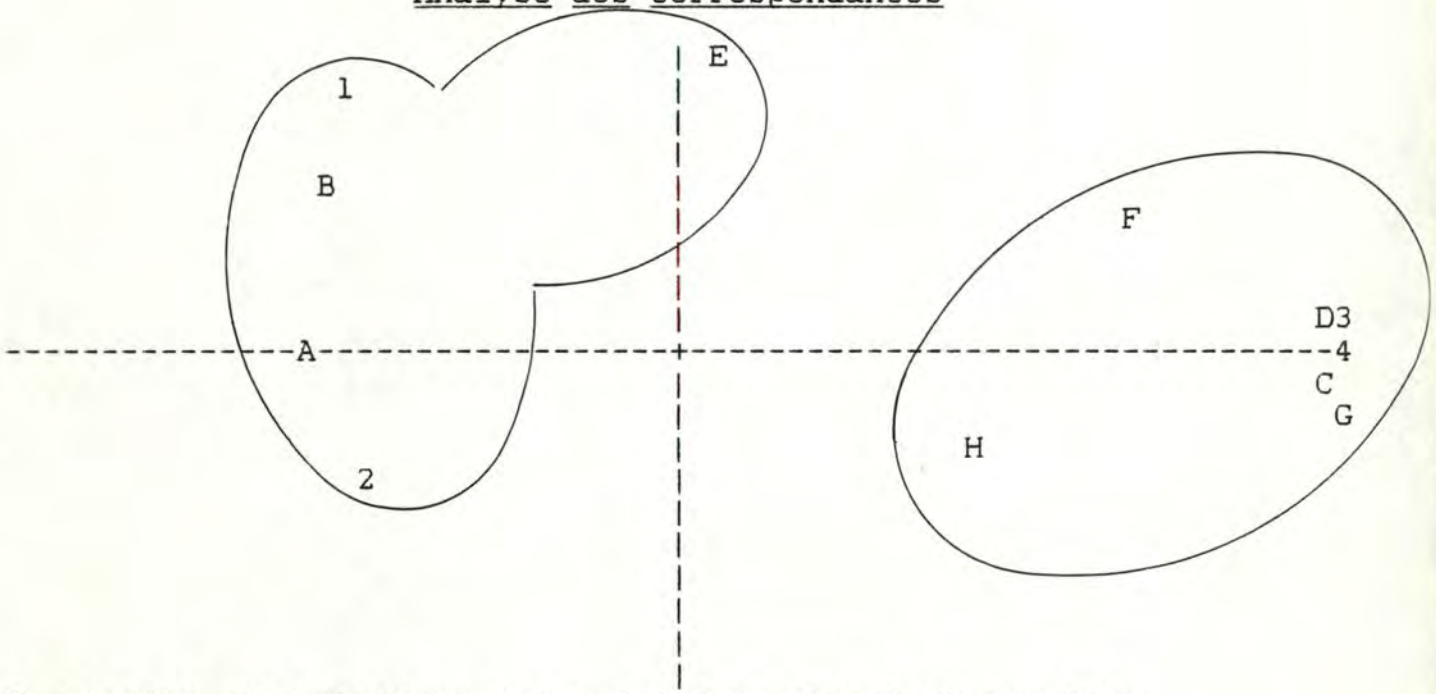
Hiérarchisation sur base de la variance



Hiérarchisation sur base de la variance



Analyse des correspondances



En réalité, le graphique est : 4.0 fois plus haut que large

1:Substitution concrète	377	2:Substitution abstraite	2510
3:Substitution partielle	1071	4:Parémie en langage clair	497
A:Proverbe indicatif	2757	B:Proverbe directif	87
C:Maxime	49	D:Aphorisme	1450
E:Locution proverbiale	44	F:Dicton	9
G:Adage	22	H:Apophtegme et autre	37

Inertie expliquée : 97.22% + 1.98% = 99.20% de l'inertie totale

Il est difficile, vu le peu de variables projetées, de donner une signification aux axes de l'analyse des correspondances. Toutefois, il ressort nettement que les adages, maximes et aphorismes sont nettement liés du point de vue de leur forme. La hiérarchisation nous permet seulement de lier aphorismes et maximes.

l'analyse de l'analogie montre très clairement la scission entre les procédés de substitution totale d'une part, de substitution partielle et de langage clair d'autre part. Le schéma ci- dessous résume les conclusions que l'on peut tirer de l'analyse entre l'analogie et la dénomination.

substitution totale (abstraite ou concrète)	langage clair substitution partielle
proverbe directif	aphorisme maxime
proverbe indicatif	dicton apophtegme et autre
locution proverbiale	adage

IV.3 LE RAPPORT ENTRE LES MOTS-CLES ET LE THEME

Puisque les mots-clés traduisent le vocabulaire courant de la parémie, il est logique qu'il existe un rapport entre le thème de la parémie et ses mots clés.

On remarque d'emblée trois points extrémaux et opposés : le malchanceux, le roi et la femme. Si on essaie d'interpréter l'axe d'inertie principal (qui explique 39.37% de l'inertie du nuage de mots), on trouve de gauche à droite : roi, lait, aieul, femme, vache, enfant, coeur (au centre), manger, pluie, nuit, mal, ruse, la chance[1], la mort et le malchanceux. L'axe principal du vocabulaire employé dans les parémies serait donc un axe de fortune --> infortune.

Le principal apport de la hiérarchisation est de parvenir à regrouper des mots d'un même noyau sémantique. On parvient ainsi à regrouper d'une manière objective les mots suivants :

1. Yeux, géniteur, individu, coeur
   Les yeux d'un individu symbolisent son caractère [rod83, pp 423]. D'autre part, le proverbe N°1280 nous dit que "le coeur, pour l'individu, c'est son propre roi".
2. Mort, ruse, malchanceux, chance[1]
3. Chemin, eau : la femme va puiser de l'eau parfois loin
4. Perdrix, père (?)
5. Nuit, pluie, mal
6. Manger, bière, pauvre, ventre
7. Aieul, lait, vache, roi : signes de fécondité, de richesse
8. Mère, viande, chien (?)
9. Femme et vérité (sic), enfant
10. Kraal et enclos : ce sont deux synonymes
11. Néant, force (souvent dans le sens de santé), pâte, bouche
12. Maison et homme

Ces regroupements sont laissés à la méditation du lecteur. Le point d'interrogation marque la scepticité de l'auteur face à un regroupement particulier. L'interprétation de ces regroupements nécessite une bonne connaissance de la culture du Burundi et s'adresse plus à un sociologue qu'à un linguiste et, a fortiori, qu' à un informaticien.

---

(1) La place de la chance à cet endroit ne doit pas surprendre Parler de la chance signifie aussi parler de la malchance

IV.4 RAPPORT ENTRE LES MOTS SYMBOLIQUES ET LE THEME

On s'attend naturellement à ce que la distribution des mots symboliques dans les différentes classes thématiques ne soit pas le fruit du hasard. Ceci se vérifie de fait en ce qui concerne les parémies du Burundi.

La hiérarchisation offre au regard du lecteur certains regroupements impposibles à saisir sans une explication symbolique adéquate. C'est le but des quelques lignes qui suivent. Le décodage des mots symboliques imprimés ci-dessous entre "(" ")" se trouve dans [rod83, pp 423-424].

1. Serpent, précipice, chien et léopard : Le serpent (inimitié), le chien (lâcheté) et le précipice (traitrise) s'opposent au thème du léopard (noblesse).

2. Rosée,pygmée,hyène et belle-mère : Le pygmée et l'hyène vivent comme des sauvages dans la brousse. Il s'agit d'êtres que l'on préfère éviter, tout comme l'on préfère éviter la belle-mère et la rosée (désagréments).

3. Maquis, jour et dent : le maquis (monde hostile) et la dent (aggresivité) s'opposent au jour (période de prospérité).

4. Tutsi, valet, gale : Le valet("Hutu") et le Tustsi constituent respectivement le bas et le haut de l'échelle sociale. La gale (pauvreté) caractérise le valet.

5. Bière, beurre : Tous deux expriment la richesse.

6. Epar et oeil : L'épar(défenseur) représente la protection face aux regards (envie, cupidité)

7. Pluie,nuit et tabac : La nuit est une période angoissante. le vent et la pluie symbolisent la gêne, les inconvénients.

8. Sel, chèvre et grenier : Le sel symbolise le superflu. La chèvre (négligence) est antonyme du grenier (richesse) signe de prévoyance.

9. Banane, baratte, rat et famine: Le rat(misère) ne connait ni baratte (richesse), ni banane(richesse) mais bien la famine(période critique).

10. Jambe, bouclier, sorgho, tambour, lait, roi, van, marteau, taureau et foudre : Le lait (richesse et protection) et le sorgho(richesse) sont des caractéristiques de la royauté. Les jambes symbolisent les démarches envers des personnalités importantes. Le bouclier et le van (protection), le marteau (pouvoir cheffal) et le tambour (règne, pouvoir) sont autant d'attributs typiques d'un roi. Celui-ci peut être incarné par la foudre (pouvoir) ou par le taureau (chef).

11. Veau et lance : La lance (homme, guerrier) s'oppose au veau (jeunesse, immaturité).

12. Marmite, bâton, suie : La suie (défauts visibles) appartient au bâton (compagne élue) ou à la marmite (femme, épouse).

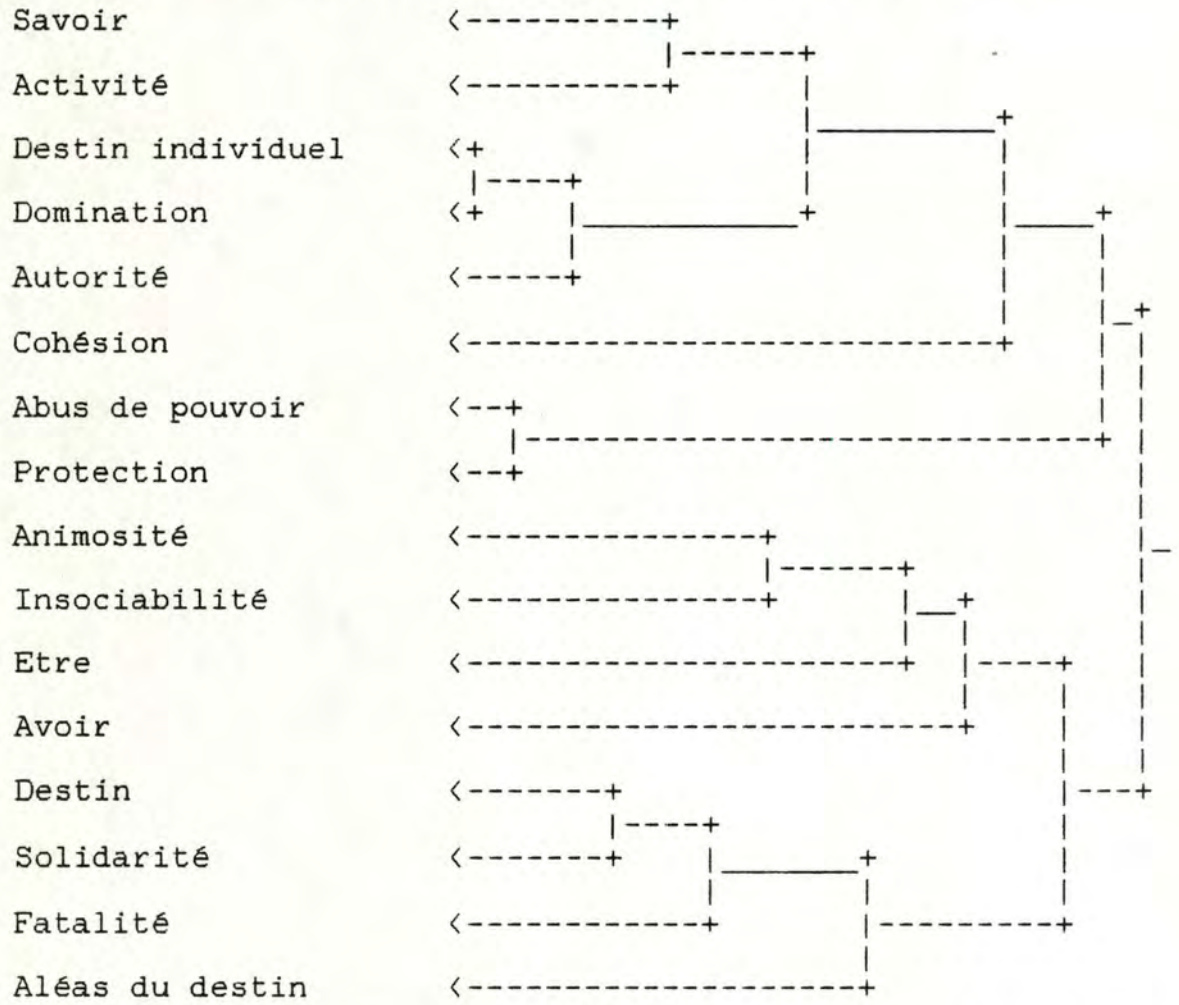
13. Meule et gendre : Meule et meuleton signifient respectivement femme et fille.

14. Proverbe, mou et fourmi : Le proverbe (traduisez parémie) est "sagesse, enseignement et vérité. Il s'oppose au mou (leurre, illusion) et à la fourmi (infériorité), un animal peu intelligent que l'on écrase du pied.

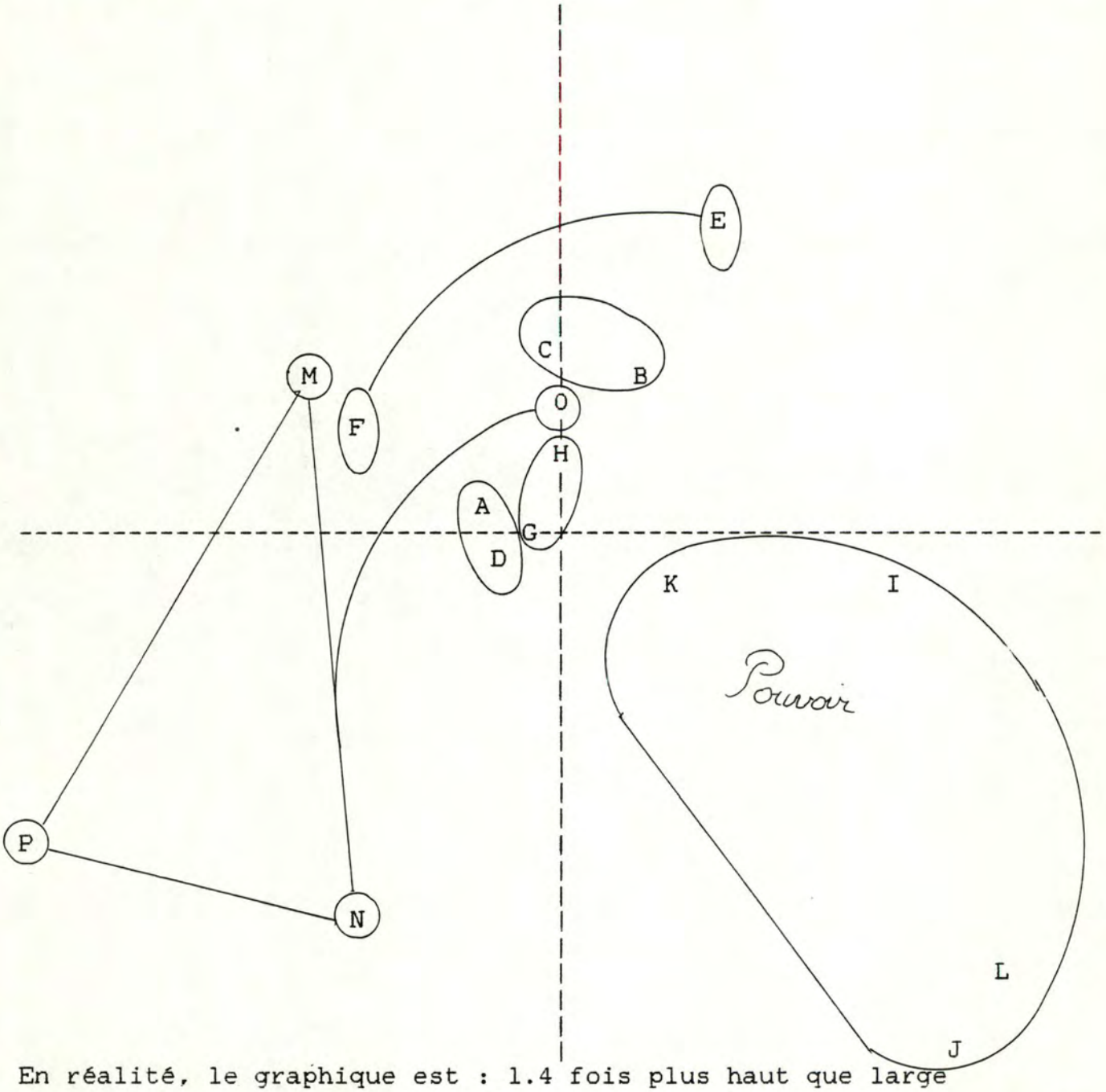
IV.5 LE THEME DE LA PAREMIE PAR RAPPORT AU VOCABULAIRE EMPLOYE.

Croiser l'aspect thématique de la parémie avec les mots clés permet d'établir un lien entre les thèmes utilisant le même profil de vocabulaire.

La hiérarchisation donne les résultats suivants :



Ceci nous révèle quelques couples de thèmes nettement liés : savoir et activité; abus de pouvoir et protection; animosité et insociabilité; destin et solidarité; fatalité et aléas du destin. Nettement plus révélateur est le graphique des correspondances ci-contre.



En réalité, le graphique est : 1.4 fois plus haut que large

A:Avoir	513	B:Activité	311	C:Savoir	416
D:Etre	471	E:Cohésion	539	F:Solidarité	165
G:Insociabilité	463	H:Animosité	303	I:Autorité	99
J:Protection	261	K:Domination	177	L:Abus de pouvoir	84
M:Destin	91	N:Aléas du destin	287	O:Destin individue	54
P:Fatalité	170				

Inertie expliquée : 39.37% + 36.11% = 75.49% de l'inertie totale

Rappelons qu'un des buts recherchés était d'éprouver la cohérence de la classification manuelle [rp intro chIII]. Cette classification thématique apparait très nettement sur le graphique des correspondances.

Le monde du pouvoir regroupe toutes ses classes dans le quadrant inférieur droit du graphique. Les trois points les plus à gauche du graphique concernent le destin. Le destin individuel, quant à lui, se trouve coincé entre ,d'une part, l'avoir et l'être et, d'autre part, l'activité et le savoir, quatre thèmes constituant le rapport de l'être au monde matériel. Le destin individuel ferait donc lui aussi partie de ce monde matériel. Le rapport de l'homme avec ses semblables se trouve très clairement éclaté en deux grands thèmes : cohésion-solidarité et animosité-insociabilité.

Remarques finales

Le lecteur s'étonnera peut-être de l'absence d'exemples liés aux critères 4,5,6 et 7. Au moment de reproduire cet ouvrage, aucune interprétation n'a été faite pour ces quatres critères. Cela ne signifie pas pour autant que ces quatre critères sont peu pertinents. Certains croisements entre ces critères se trouvent en annexe D afin de permettre au lecteur féru de parémiologie de se faire son idée personnelle. L'interprétation que j'ai voulu détailler ici a été établie avec l'aide du R.P. Rodegem, auteur de "Paroles de sagesse au Burundi"[rod83]. Elle possède certes une part de subjectivité, mais l'auteur a été spécialement soucieux dans sa démarche interprétative de la plus grande rigueur possible. Certaines interprétations douteuses, quoiqu'exaltantes, n'ont pas trouvé place ici.

Le but de ces interprétations était de convaincre le lecteur de la puissance des méthodes employées et de la pertinence des critères retenus. Il va de soi que le plus gros du travail d'interprétation et de dépouillement des résultats relève plus de la compétence du parémiologue que de celle l'informaticien.

CHAPITRE V

EXTENSIONS POSSIBLES ET CONCLUSIONS

V.1 EXTENSIONS POSSIBLES

V.1.A VS L'ANALYSE SYNTAXIQUE

Comme détaillé dans le chapitre II, l'analyseur syntaxique développé était correct mais incomplet. Il s'agissait en fait d'un prototype.

Il est certes possible d'améliorer les performances de l'analyseur. Pour cela, il faudrait faire une analyse un peu plus en profondeur de la syntaxe du rundi et spécialement des contractions possibles entre les différentes parties du mot. Une autre possibilité envisageable serait de procéder d'une manière incrémentale. On a en effet vu que la grande faiblesse de l'analyseur syntaxique était qu'il ne connaissait pas les différents thèmes des verbes et des substantifs. Il est par contre relativement aisé de conjuguer ou de décliner un mot lorsqu'on en connaît le thème.
On pourrait procéder en plusieurs phases :

1. Construire un index vide de thèmes .

2. Parcourir le fichier du Kirundi et inscrire dans l'index des thèmes ceux qui sont sûrs (thème d'un mot-clé ou thème très courant). L'analyseur syntaxique intervient dans cette phase.

3. Parcourir le fichier du Kirundi en examinant , pour chaque mot, la possibilité pour celui-ci de provenir d'un thème conjugué ou décliné à partir du fichier des thèmes.

4. Garder dans le fichier du kirundi uniquement les mots non identifiés (ceci peut se faire lors des phases deux et trois)

5. Si, lors des phases deux et trois, aucun mot du fichier en Kirundi n'a pu être identifié, alors il faut stopper , sinon recommencer à partir de la phase deux.

Il ne faut pas perdre de vue qu'il ne s'agit pas dans le cas présent de construire un analyseur de la langue Kirundi, mais "simplement" un analyseur de ± 8000 mots contenus dans des parémies.

Sans tenir compte des accents, on trouve qu'il y a environ huit mille mots qui sont formellement différents. Parmi ceux-ci se trouvent les verbes. Etant donné que ceux-ci se conjuguent de manière extrêmement variée, il est très probable que le nombre de mots différents devrait encore certainement se réduire.

V.1.B VS ANALYSE GRAMMATICALE

L'analyse grammaticale en est restée à un point très sommaire. En effet, elle se base uniquement sur la présence de telle ou telle particule pour classifier une parémie donnée. Il va de soi qu'une meilleure connaissance des variations possibles du mot permettrait de diminuer le nombre des inclassés. L'idéal serait de l'amener à un seuil où la classification manuelle serait possible. Une autre solution serait peut-être de considérer la traduction française, ce qui présente l'avantage de ne pas devoir posséder un tant soit peu la grammaire du Kirundi.

V.1.C AUTOMATISATION DE LA CLASSIFICATION MANUELLE.

Cela permettrait un contrôle de validité des données codées nettement plus performant. L'établissement du critère 1 pourrait être basé sur la détection de mots-clés ou de mots symboliques. Les critères quatre et cinq me semblent eux-aussi automatisables dans une certaine mesure. En ce qui concerne les critères deux et trois, une automatisation, même relativement fiable, me semble relever de la plus pure utopie.

V.1.D ANALYSE DE PARÉMIES D'AUTRE CULTURES.

En concevant mes programmes, j'ai toujours essayé dans la mesure du possible de séparer ce qui dépendait des particularités du rundi de ce qui en était indépendant. Ainsi par exemple, le classement selon le sixième critère pourrait tout aussi bien concerner une autre langue, exception faite des quelques règles de prononciation détaillées dans le titre II.1

Il va de soi que les programmes d'établissement des critères sept et huit sont tout à fait dépendants de l'analyseur syntaxique. Mais il est extrêmement aisé de modifier cet analyseur en changeant les structures syntaxiques et les grammaires transformationnelles. Pour une modification aisée, j'ai choisi de créer des fichiers contenant tous ces renseignements. Ainsi, une modification apportée à la grammaire transformationnelle ou aux déclinaisons et conjugaisons des mots se répercute directement dans le comportement de l'analyseur syntaxique, sans qu'il soit nécessaire d'apporter la moindre modification au programme pascal qui l'abrite.

V.2 CONCLUSIONS

Au cours de cette conclusion j'essaierai d'évaluer d'une manière critique le travail accompli.

La question de départ " Est-il possible d'appliquer des méthodes d'analyse automatique à des parémies ? " trouve selon moi une réponse fondamentalement positive, même s'il convient d'y apporter certaines précisions.

Si certains résultats paraissent probants, il faut se garder de conclure trop vite que l'ordinateur est capable d'analyser, voire de "comprendre" des citations sentencieuses.
Et ceci pour plusieurs raisons.

a) L' ordinateur n'a pas analysé des parémies mais un type de représentation formelle de celles-ci. Par le codage, la parémie gagne en manipulabilité, mais elle perd aussi pas mal de ses caractéristiques, notamment son contexte. Toutefois, les résultats montrent que l'analyse multi-critères est capables de mettre en évidence des corrélations cohérentes dans l'ensemble.

b) Cela ne signifie évidement pas que les méthodes développées dans ce mémoire sont capables de mettre en évidence les traits fondamentaux des parémies, mais bien de mettre en évidences certaines corrélations qui se vérifient dans la réalité.

c) Un autre aspect de la démarche d'analyse présente un danger de type tautologique. On se souvient en effet des conclusions tirées à propos du rapport entre structure analogique et dénomination. La machine ne prouve rien. Elle met "simplement" en évidence des corrélations que l'opérateur humain établit plus ou moins consciemment lors de sa classification. Toutefois, il faut bien admettre que certains critères sont relativement objectifs. C'est le cas, notament, pour le thème du proverbe, son vocabulaire, sa structuration analogique , et les mots symboliques qu'il emploie.

L'ouvrage réalisé, même s'il présente un caractère innovateur, n'est qu'un premier pas dans une direction jusqu'ici fort peu explorée. J'espère que d'autres parémiologues et informaticiens pousseront plus loin des recherches pouvant aider à une meilleure compréhension de ces Paroles de Sagesse qui nous enseignent un "mieux vivre".

TABLE DES MATIERES

[Ait84] Aït Hamlat A., Benzéeri J.P. "Analyse des répétitions et indexation automatique des documents", Les cahiers de l'analyse des données, Dunod, Paris, 1984.

[Ber81] Bertier P., Bouroche J.M. "Analyse des données multi-dimmensionnelles", Systèmes-Décisions, Presses Universitaires de France, 3° édition, Paris, 1981.

[Cho69] Chomsky N., "Structures syntaxiques", collection Points, Editions du Seuil, Paris, 1969.

[Eis78] Clema V.C., Smith B.T., Boyle J.M., Dongarra J.J., Garbow B.S., Ikebe Y., Moler C.B.
"Matrix Eigensystem Routines - Eispack Guide", Lecture Note in Computer Science, Springer-Verlag, 2° édition, Berlin, 1978.

[Leb79] Lebart L., Morineau A., Fénélon J.P. "Traitement des données statistiques", Méthodes et Programmes, Dunod, Paris, 1979.

[Mou72] Mounin G. "Clefs pour la sémantique", Clefs, Seghers, Paris, 1972.

[Per83] Perrot J. "La Linguistique", Que Sais-Je, Presses Universitaires de France, 2° édition, Paris, 1983.

[ren83] Reinert A. "Une méthode de classification descendante hiérarchique ; application à l'analyse lexicale par contexte", Les cahiers de l'analyse des données, Dunod, Paris, 1983.

[Rod83] Rodegem F. "Paroles de Sagesse au Burundi", Peeters, Leuven 1983.

[Rod67] Rodegem F. "Précis de grammaire Rundi", Editions Scientifiques Story-Sciencia, 1967.

Facultés Universitaires Notre-Dame de la Paix

Année académique 1984-85

Elaboration et application de méthodes multi-critères d'analyse automatique pour 4456 parémies du Burundi

A N N E X E S

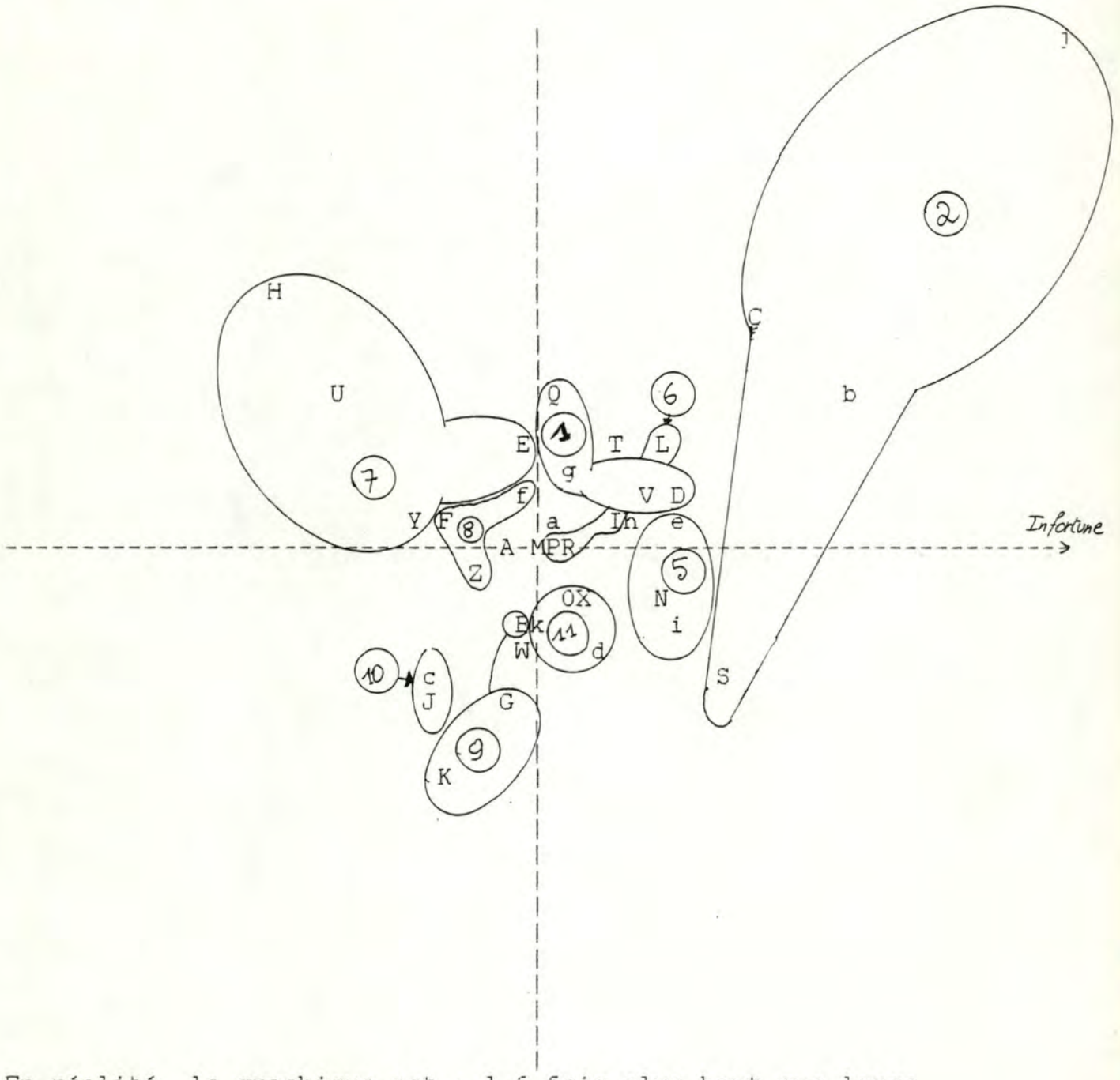
Par
Jean-Marc Dinant

Mémoire présenté en vue de l'obtention du grade de licencié et maître en informatique

Promoteur : Professeur Jean Fichefet

A D D E N D A .

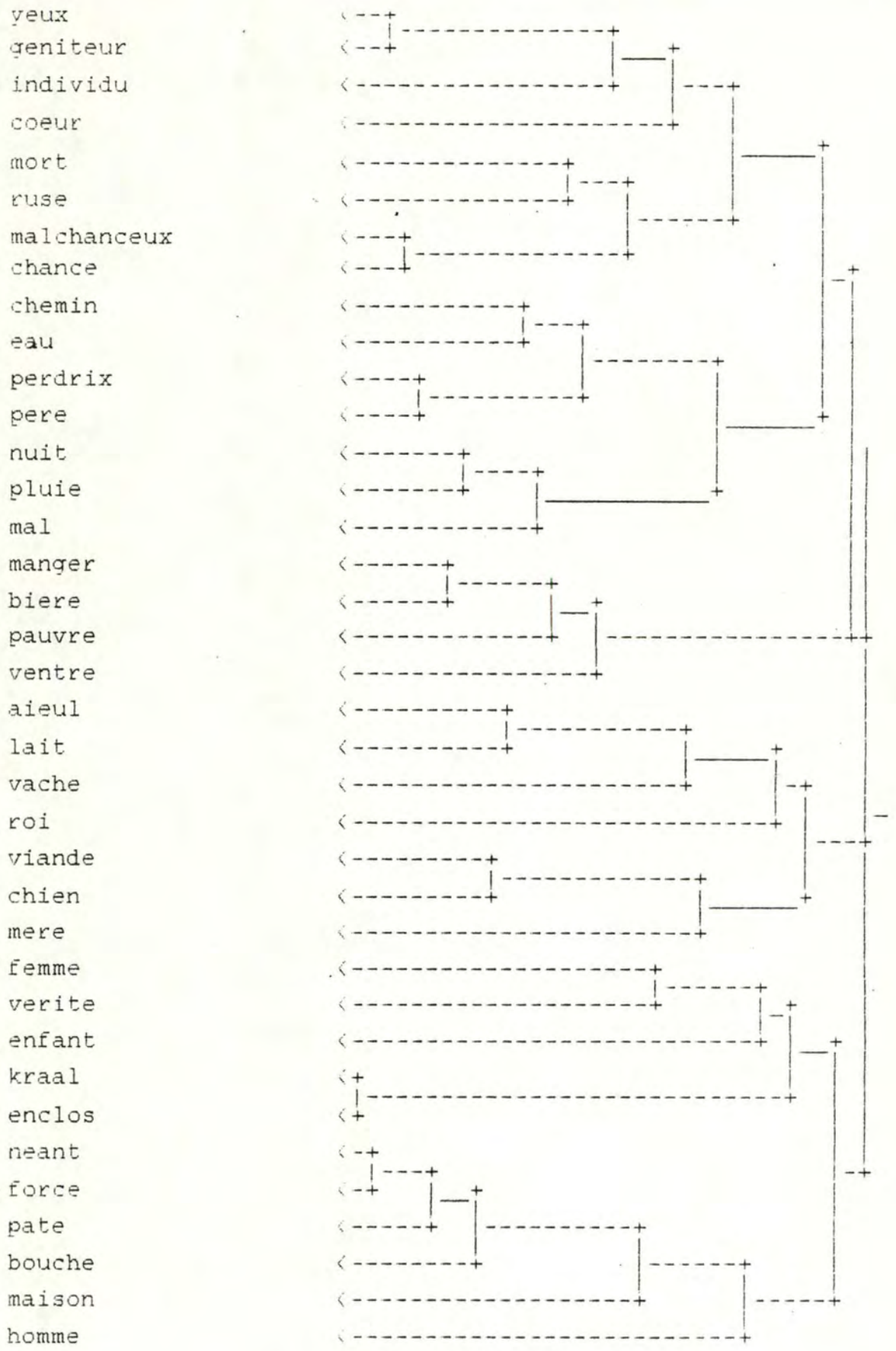
Les quatres graphiques ayant été commentés aux pages 60, 61, 62 se trouvent aux quatre pages suivantes

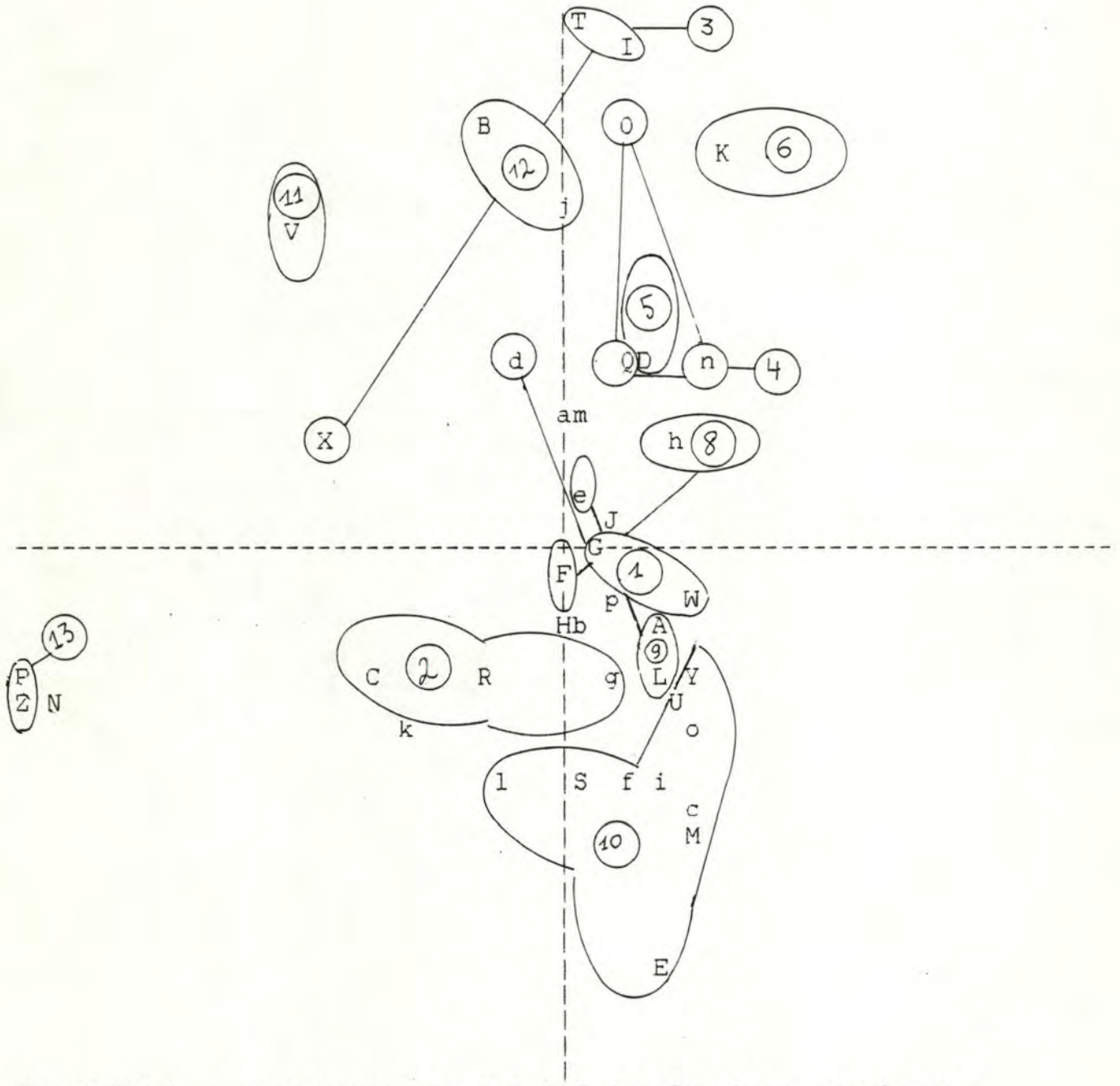


En réalité, le graphique est : 1.6 fois plus haut que large

A:homme	461	B:enfant	484	C:chance	31	D:individu	198
E:vache	204	F:chien	101	G:verite	106	H:roi	225
I:ventre	272	J:enclos	117	K:femme	158	L:pauvre	49
M:eau	121	N:mal	148	O:pate	81	P:biere	70
Q:coeur	100	R:manger	75	S:ruse	19	T:chemin	70
U:lait	86	V:geniteur	56	W:maison	147	X:bouche	77
Y:aieul	57	Z:mere	255	a:pere	60	b:mort	71
c:kraal	63	d:force	54	e:pluie	38	f:viande	72
g:yeux	66	h:perdrix	53	i:nuit	84	j:malchanceux	18
k:neant	57						

Inertie expliquée : 39.37% + 36.11% = 75.49% de l'inertie totale

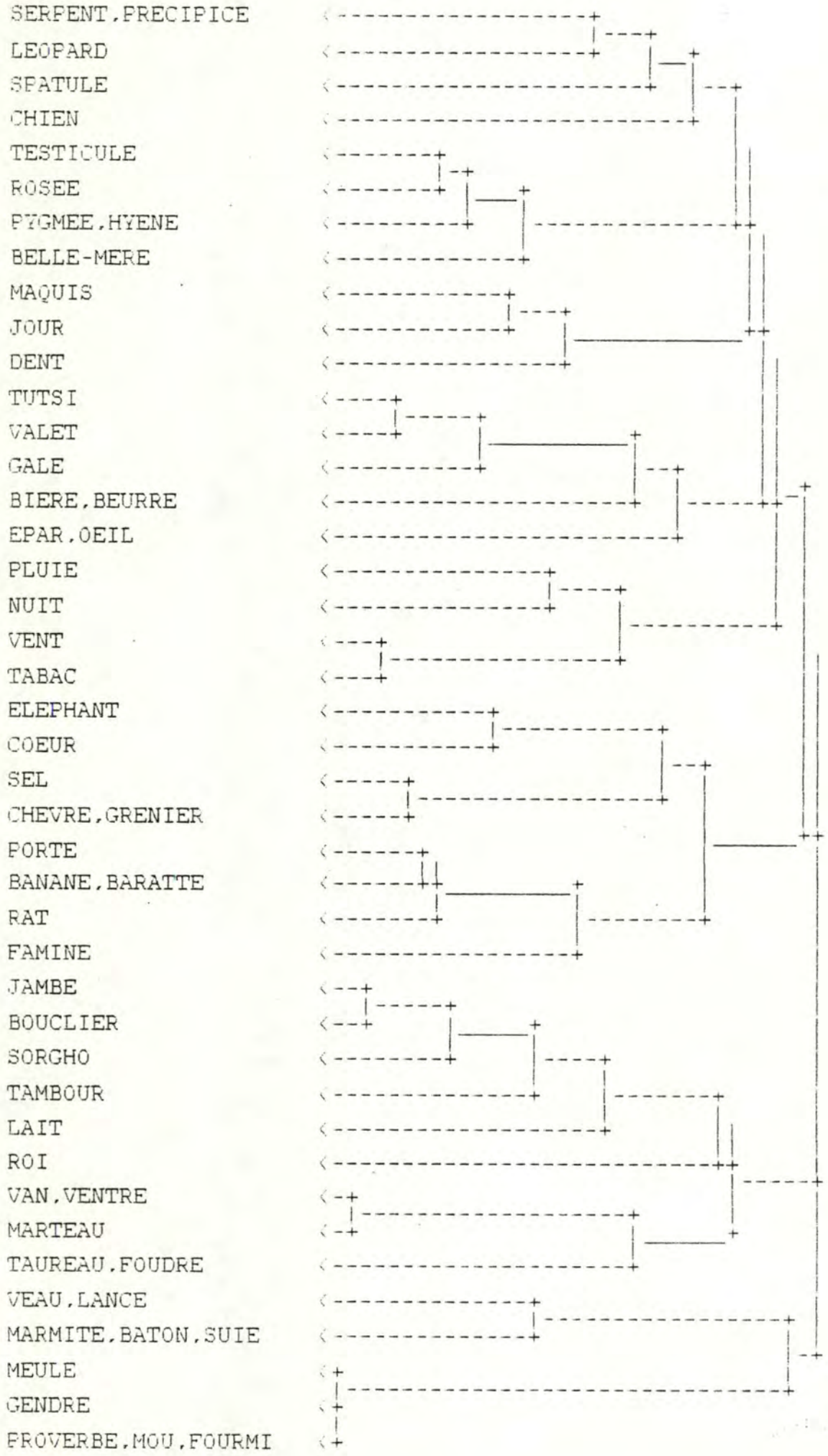

yeux
geniteur
individu
coeur
mort
ruse
malchanceux
chance
chemin
eau
perdrix
pere
nuit
pluie
mal
manger
biere
pauvre
ventre
aieul
lait
vache
roi
viande
chien
mere
femme
verite
enfant
kraal
enclos
neant
force
pate
bouche
maison
homme



En réalité, le graphique est : 1.1 fois plus haut que large

A:BANANE,BARA	8	B:MARMITE,BAT	23	C:BELLE-MERE	9	D:BIERE,BEURR	21
E:BOUCLIER	3	F:CHEVRE,GREN	20	G:CHIEN	59	H:COEUR	9
I:DENT	12	J:ELEPHANT	8	K:EPAR,OEIL	14	L:FAMINE	13
M:TAUREAU,FOU	14	N:PROVERBE,MO	20	O:GALE	12	P:GENDRE	2
Q:VALET	7	R:PYGMEE,HYEN	16	S:JAMBE	9	T:JOUR	5
U:LAIT	22	V:VEAU,LANCE	12	W:LEOPARD	4	X:MAQUIS	6
Y:MARTEAU	3	Z:MEULE	2	a:NUIT	16	b:PLUIE	18
c:PORTE	3	d:SERPENT,PRE	7	e:RAT	8	f:ROI	54
g:ROSEE	4	h:SEL	5	i:SORGHO	9	j:SPATULE	3
k:TABAC	3	l:TAMBOUR	13	m:TESTICULE	5	n:TUTSI	4
o:VAN,VENTRE	9	p:VENT	7				

Inertie expliquée : 38.13% + 32.26% = 70.40% de l'inertie totale


SERPENT,PRECIPICE
LEOPARD
SPATULE
CHIEN
TESTICULE
ROSEE
PYGMEE,HYENE
BELLE-MERE
MAQUIS
JOUR
DENT
TUTSI
VALET
GALE
BIERE,BEURRE
EPAR,OEIL
PLUIE
NUIT
VENT
TABAC
ELEPHANT
COEUR
SEL
CHEVRE,GRENIER
PORTE
BANANE,BARATTE
RAT
FAMINE
JAMBE
BOUCLIER
SORGHO
TAMBOUR
LAIT
ROI
VAN,VENTRE
MARTEAU
TAUREAU,FOUDRE
VEAU,LANCE
MARMITE,BATON,SUIE
MEULE
GENDRE
PROVERBE,MOU,FOURMI

A N N E X E S

" Qui est passé en contrebas d'une maison
ne sait pas ce qui s'y trouve. "

Proverbe du Burundi.

ORGANISATION DE L'ANNEXE

L'annexe qui suit possède cinq parties numérotées A, B, C, D, E.

[A] La découpe en module (généralités; découpe des modules bt matop, string ; flux des données)

[B] La liste des programmes sources : d'abord les modules, ensuite les programmes principaux utilisant ces modules. Chacune de ces deux parties est classée par ordre alphabétique croissant du nom des modules/programmes.

[C] La liste des classes utilisées avec pour chaque classe : son numéro, le numéro du critère qu'elle concerne, son nom , son effectif, son code minimum et maximum

[D] Certains résultats probants : certaines tables de contingence entre deux critères quelconques classées selon N. N est un nombre à deux chiffres n1, n2, chacun de ces deux chiffres identifiant un critère de 1 à 9. Deux tables ⟨n1n2⟩ ⟨n2n1⟩ sont redondantes. Une seule seulement de ces deux tables se trouvera en annexe : celle pour laquelle le nombre de classes de n1 est supérieur au nombre de classes de n2[1]. A la suite des tables de contingence présentes dans l'annexe, le lecteur trouvera la hiérarchisation suivie, si cela semble opportun, d'une analyse des correspondances.

[E] Cette annexe contient des formules et explications mathématiques.

Remarque : numérotation des pages.

Chaque page d'annexe comporte

1. En haut : la lettre de l'annexe, son titre détaillé, le numéro de la page en question dans l'annexe à laquelle elle appartient.

2. En bas : son numéro dans le volume d'annexe.

(1) Ceci est motivé par une question de mise en page.

1 GENERALITES ET PRINCIPES

Décrire la découpe modulaire suppose au préalable la définition de plusieurs types abstraits. Le premier a déjà été détaillé dans le chapitre I. Il s'agit de la phrase.

Le deuxième type abstrait est la classe. Cette notion de classe a déjà été définie dans le chapitre II.

Une table de contingence est un tableau de nombres entiers. La taille maximum[3] de ce tableau est 300.

Enfin, la matrice est un type abstrait qui peut notamment représenter une table de contingence. Il s'agit d'un tableau de nombres réels[2]. La taille maximum[3] de ce tableau est 50. Un code est la suite des valeurs d'un critère d'une parémie.

Chacun de ces types abstraits ne peut être manipulé directement par une suite d'instructions en Pascal. La manipulation se fait uniquement via des primitives établies sur ces types. Une primitive est une fonction élémentaire, qu'il n'est donc pas intéressant de décomposer en plusieurs étapes.

Un module regroupe les primitives effectuables sur un type abstrait.

Chaque module gère seul le(s) type(s) abstrait(s) qu'il abrite. Il assure lui-même le contrôle de cohérence des opérations demandées ainsi que la validité des résultats transmis. Si une fonction demandée ne peut être réalisée pour une raison ou pour une autre, il le signalera à l'utilisateur en indiquant le nom du module et de la fonction où l'erreur a été détectée, et le type d'erreur rencontré.

EXEMPLE

> Un appel à la fonction "pm" demande au module matop d'effectuer une multiplication matricielle entre A et B. Or il se fait que le nombre de colonnes de A est différent du nombre de lignes de B. La multiplication ne peut donc être effectuée correctement et le module matop envoie à l'utilisateur le message suivant :
>
> FATAL ERROR [during Pm in MATOP] incompatible matrix's size

(1) C-à-d le nombre maximum de classes, tous critères confondus.
(2) Un nombre entier est a fortiori un nombre réel.
(3) Ces constantes sont facilement changeables.

On distinguera deux types d'erreurs :

1. Les erreurs FATALES : ce sont celles qui rendant la continuation de de l'exécution du programme inutile. Toutefois, le programme ne sera pas interrompu pour autant: c'est l'utilisateur lui-même qui doit juger en temps réel s'il doit ou non stopper le programme. Ce type d'erreur entraîne parfois de lui-même un arrêt du programme (par exemple suite à une division par zéro)

2. Les AVERTISSEMENTS : ce sont des erreurs qui rendent la validité du programme suspecte. Ces erreurs n'empêchent en rien la continuation du programme, mais l'opération suspecte est signalée à l'utilisateur.

Les opérations de base sur un type abstrait sont

1. La création d'une de ses occurences.
2. La modification d'une de ses occurences.
3. La suppression d'une de ses occurences.
4. L'écriture d'une de ses occurences sur un écran ou du papier
5. La lecture d'une occurence dans un fichier
6. Le sauvetage d'une occurence dans un fichier

Le module "STRING" comprend les fonctions définies sur les phrases. Il a déjà été détaillé lors du chapitre I.

Le module "MATOP" gère les primitives définies sur les matrices.

Le module "BT" prend en charge les opérations concernant les tables de contingence des classes.

Remarque

> Un module gère une occurrence particulière d'un type abstrait, non pas le type abstrait lui-même. Pour prendre un exemple quelque peu poétique, le type abstrait est à la lumière ce qu'une lampe est à son occurrence. On peut construire, transformer, stocker, montrer, regarder ou même détruire des lampes. Malgré la destruction d'un certain nombre de lampes, le concept de la lumière en lui-même n'en continue pas moins d'exister.

De tout ce qui précède, découle la structure de l'architecture modulaire. Remarquons préalablement que la parémie possède déjà 4555 occurrences dotées d'une existence définitive. La création ne se justifie donc pas pour ce type abstrait.

2 LE MODULE MATOP[1]

Les opérations de base disponibles sont:

C R E A T I O N

a) function diag(n:integer):matrix;
crée une matrice unite de dimension n x n
Erreur : si n<=0 ou n>max_size

b) procedure create_matrix (var
a:matrix;n,m:integer;mat_n:mat_nam);
crée une matrice de dimension n x m avec
tous les elements egaux a 0.
le nom de la matrice est mat_n.

Erreur : Si n, le nombre de lignes de a,
est supérieur à max_size
Si m, le nombre de colonnes de a,
est supérieur à max_size

L E C T U R E

c) procedure read_matrix(var a : matrix;name :mat_nam);

lit une matrice se trouvant sur un fichier de type text ;
name est le nom du fichier contenant la matrice
Erreur : Si la matrice porte le nom d'un fichier inexistant
Si le nombre de lignes de a est supérieur à max_size
Si le nombre de colonnes de a est supérieur à max_size

M O D I F I C A T I O N

d) function pm(a,b: matrix) : matrix ;
calcule le Produit Matriciel A X B
Erreur : Si les tailles de A et B sont incompatibles
c-à-d si le nombre de colonnes de A est différent
du nombre de colonnes de B

e) function sum(a,b:matrix):matrix;
calcule la somme de deux matrice de même taille
Erreur : Si les tailles sont différentes

f) function ps(c:elem_type;a:matrix):matrix;
calcule le produit scalaire de ca

g) function trans(a:matrix):matrix;
calcule la transposee de la matrice A

---

(1) MATrix's OPerations

h) procedure totalize (var k:matrix);
ajoute a la matrice k une ligne et une colonne de totaux

i) procedure kill_zeroes(var a :matrix);

supprime les lignes et les colonnes de zeros d'une matrice a en la compactant

j) procedure kill_col(var a:matrix;n:integer);

supprime la ligne n de la matrice a
message d'erreur : ssi la ligne n de a n'existe pas

k) procedure kill_line(var a:matrix;n:integer);
supprime la colonne n de la matrice a
message d'erreur : ssi la colonne n de a n'existe pas

l) procedure class_vp(var l,u : matrix);
classe les valeurs propres et les vecteurs propres par ordre croissant
des valeurs propres ; les vecteurs propres sont les colonnes de u

m) procedure vp(a:matrix;var vap,vep : matrix);
calcule les vecteurs et valeurs propres de la matrice A

n) procedure verify (a,l,x:matrix);
verifie si l[i]x[i] = ax[i] ,autrement dit
si l et x sont respectivement valeurs et vecteurs propres de a

o) procedure cp1(k:matrix;var z,lambda : matrix);
effectue l' analyse des correspondance pour les lignes de la matrice k
Z contiendra en résultat les coordonnées des projections (1 colonne par point)
la première colonne de lambda contient les valeurs propres
la deuxième ligne la somme marginale de ces valeurs

p) procedure cp2(k:matrix;var z,lambda : matrix);
semblable à cp1, mais projette les points-lignes

E C R I T U R E

q) procedure print_matrix(a:matrix;ent,nbdec:integer);
imprime la matrice a à l'écran.
ent: nombre maximum de caractères écrits par nombre ttc
nbdec : nombre de decimales par nombre
Erreur : Si une ligne de chiffres dépasse 120 caractères, Elle sera imprimée en plusieurs lignes de 120 caractères

3 LE MODULE BT[1]

Les primitives définies sur les classes sont

L E C T U R E

a) procedure read_clas;

Pour chaque classe, cette procedure lit dans le fichier clfile.dat

1°) le nom de la classe (sur une ligne) 2°) sa borne inferieure ,sa borne superieure et son effectif (sur la même ligne) 3°) le numéro de la modalité a laquelle elle appartient

C R E A T I O N

b) procedure create_clas(nom : class_name ; bi,bs,effectif,zone : integer) ;

insère une nouvelle classe dans la liste des classes (pas sur fichier !) la classe ajoutée sera la dernière du critère

S U P P R E S S I O N

c) procedure kill_clas(nbc : integer);

supprime la nbcième classe du tableau des classes (pas du fichier des classes !)

E C R I T U R E

d) procedure print_clas;

imprime la liste des classes et leurs caractéristiques

S A U V E T A G E

e) procedure save_clas; Pour chaque classe, cette procedure écrit dans le fichier clfile.dat

1°) le nom de la classe (sur une ligne)
2°) sa borne inferieure ,sa borne superieure et son effectif (sur la même ligne)
3°) le n° de la modalité a laquelle elle appartient

(1) B(uild) T(able)

Primitives définies sur les tables de contingence :

C R E A T I O N

a) procedure build_clas(var frequency : big_matrix ; zonel,zone2 : integer);

calcule les effectifs resultant de l'intersection de la zonel et de la zone2 ; autrement dit, crée une table de contingence entre le critère zonel et le critère zone2.

M O D I F I C A T I O N

b) procedure reduce(var frequency : big_matrix; zonel,zone2 : integer);

supprime les lignes et les colonnes nulles de frequency répercute la réduction sur les classes mais pas dans le fichier. Autrement dit supprime les classes d'effectif nul.

4 MODULE STRING

Les primitives définies sur les mots et sur les phrases sont

C R E A T I O N D ' U N E P H R A S E

1. procedure desaccentuer(a:phrase ; var b : phrase);

   a : parémie en rundi a traiter (les 10 premieres lettres sont un identifiant non pris en ligne de compte)

   b : contiendra la phrase en rundi sans les accents b ::= ‹' '›‹mot› ... ‹' '›‹mot›‹' '› ‹mot› ::= ‹suite non-vide de lettres minuscules›

   en outre convertit toute majuscule en minuscule

   quatres sortes de caracteres peuvent se presenter : un accent "pur" un backspace une lettre sans accent une lettre accentuee

D E C O M P O S I T I O N D ' U N E P H R A S E E N M O T S

2. procedure dcm(a : phrase) ;

   DéCoMpose la phrase a en une série de mots w[1], w[2], ... , w[nbw] un mot = une suite contigüe de caractères non blancs

M O D I F I C A T I O N

3. procedure capitalize (var a : phrase);

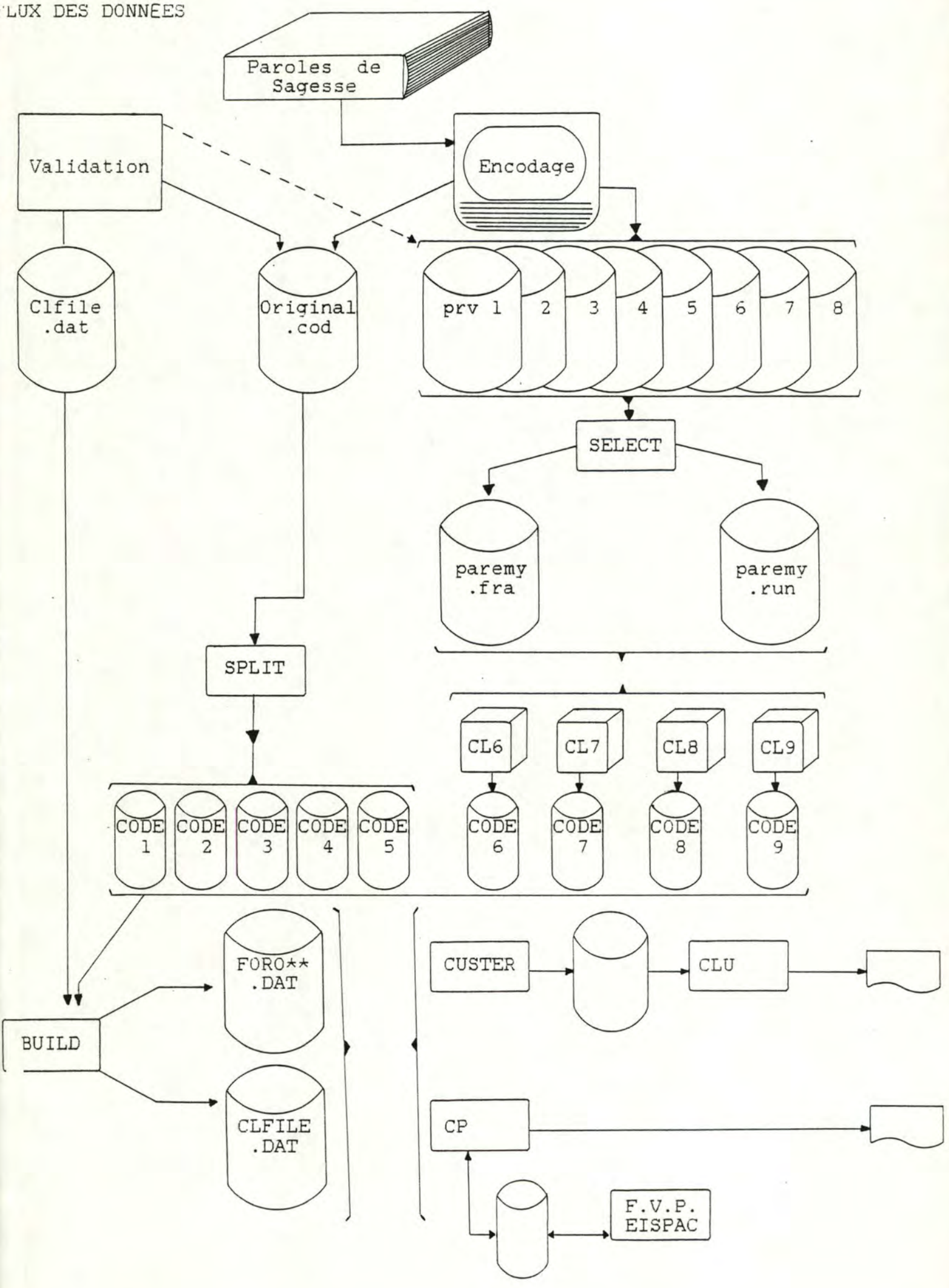
   transforme les lettres de a en lettres majuscules

E C R I T U·R E

4. procedure print_phrase(a : phrase);

   écrit une phrase de manière à avoir une représentation agréable a l'oeil ; utilise deux lignes pour écrire une phrase

FLUX DES DONNEES



5 FLUX DE DONNEES.

Le schéma suivant illustre le flux des données entre les différents programmes et résume tout le travail d'analyse des parémies qui a été accompli.

La symbolique employée est la suivante :

1. Un cylindre représente un fichier.
2. Un rectangle représente un (ou plusieurs) programme(s).
3. Un cube signifie un programme de classement associé à un numéro de critère.
4. Une feuille est un résultat sur papier.
5. Une flèche illustre un flux de données dans un certain sens.

Au début du travail, les codes des cinq critères se trouvaient sur un seul fichier. Le programme split éclate ces cinq codes en cinq fichiers codes séparés de 4455 lignes. Chaque ligne est un nombre entier écrit sur trois caractères. "C11.res" contient les codes du critère 1, "c12.res", les codes du critère deux, etc ... . Le rundi, le français et l'explication se trouvent mélangés dans 8 fichiers séparés. Le programme "select" a pour effet de créer un fichier unique avec toutes les parémies en rundi ("paremy.run") et un autre fichier ("paremy.fra") avec leur traduction.

Les programmes C16, CL7, CL8, CL9 créent des fichiers de codes pour les quatre nouveaux critères. Ces fichiers codes peuvent posséder plusieurs chiffres par ligne mais comprendront toujours 4455 lignes.

L'analyste génère le fichier des classes "clfile.dat" qui contient les indications pour effectuer les partitions.

Le programme "Build" crée la table de contingence entre deux critères i et j. Cette table de contingence se compose d'une matrice sauvée dans le fichier "for<i><j>.dat". En outre le programme build supprime les classes vides et enlève les lignes correspondantes (nulles) de la table de contingence. L'analyse des correspondances ne supporte pas, en effet, de lignes nulles.

A ce stade, l'analyse est possible via cluster (hiérarchisation) ou cp (analyse des correspondances). "clu" est un programme Pascal qui embellit le graphique de la hiérarchisation. Pour l'analyse de correspondances, le calcul des valeurs/vecteurs propres se fait par le logiciel Eispack © écrit en Fortran IV. Le passage des données/résultats entre Pascal et Fortran se fait par fichier.

```
[environment ('bt.env')]

module bt(output,clfile);

const   max_zone = 9;
        { maximum number of category (not classes) of paremy
          a paremy always one class in each category=zone    }

        max_clas = 300 ;
        {  nombre maximum de classes  }

        max_par = 5000 ;
        {  nombre maximum de paremies  }


type    class_name = varying[30] of char;

        typclas = record
                bi,bs    : 0..max_par ;
                effectif: 0 .. max_par;
                zone     : 1..max_zone ;
                name     : class_name ;
        end; {of record}

        class_range = 1.. max_clas ;

        big_matrix = array[1..max_clas,1..max_clas] of integer ;

var     class      : array[1..max_clas] of typclas ;

       { contiendra toutes les classes toutes modalites confond

        nb_clas     : class_range ;
        { nombre de classes reellement utilisees  }

        clfile     : text;
        { contient la definition de chaque classe }


       ref      : packed array[1..max_zone,0..max_par] of class_range
        {  ref[a,i] contient le numero de la classe de la zone a
           qui contient le code i  }
```

```
[global]procedure read_clas;

{  Pour chaque classe, read_clas lit
   dans le fichier clfile.dat

        1°) le nom de la classe (sur une ligne)
        2°) sa borne inferieure ,sa borne superieure
            et son effectif (sur la même ligne)
        3°) le numéro de la modalité a laquelle elle appartient  }

var     i , j   : integer ;

begin

       for i:=1 to max_zone do for j:=0 to max_par do ref[i,j] := 0
        nb_clas:=0;
        open(clfile,file_name:='clfile.dat',history:=readonly);
        reset(clfile);
        while not eof(clfile)
        do begin
                nb_clas:=nb_clas+1;
                with class[nb_clas]
                do begin
                        readln(clfile,zone);
                        readln(clfile,bi,bs,effectif);
                        readln(clfile,name);
                        for i:=bi to bs
                        do if ref[zone , i] = 0
                           then ref[zone,i]:=nb_clas
                           else writeln('La classe ',nb_clas:1,
                                        ' en chevauche une autre');
                    end ; {  of with  }
           end;
        close(clfile);
end;

[global]procedure save_clas;
{  Pour chaque classe, save_clas écrit dans le fichier clfile.dat

        1°) le nom de la classe (sur une ligne)
        2°) sa borne inferieure ,sa borne superieure
            et son effectif (sur la même ligne)
        3°) le numéro de la modalité a laquelle elle appartient  }

var     i : integer ;

begin
        open(clfile,file_name:='clfile.dat',history:=new);
        rewrite(clfile);
        for i:=1 to nb_clas
        do begin
           writeln(clfile,class[i].zone);
           writeln(clfile,class[i].bi,class[i].bs,class[i].effectif
           writeln(clfile,class[i].name);
           end;
        close(clfile,save);
end;
```

```
[global] procedure create_clas(nom : class_name ;
                               bi,bs,effectif,zone : integer ) ;

{  insère une nouvelle classe dans la liste des classes
   (pas sur fichier !)  }
{  la classe ajoutée sera la dernière du critère  }

var     i        : integer ;
        presence : integer ;
                   {  numéro de la classe devant laquelle insérer

                                la nouvelle classe créée = n° de la n

begin
        if nb_clas >= max_clas
        then writeln('erreur in create_class')
        else begin
                presence := 0 ;
                for i:= 1 to nb_clas
                do if class[i].zone <= zone
                   then presence := i ;

                {  presence = max (i ! class[i] <= zone )    }

                presence := presence + 1;

                if presence = 1
                then if zone = 9 then presence := nb_clas + 1

                                 else writeln('unable to create class

                while ((class[presence - 1].zone = zone)
                   and (class[presence - 1].bs   >  bi )
                   and (presence > 1 ))
                do presence := presence - 1 ;


                for i:= nb_clas downto presence
                do class[i+1]:=class[i] ;

                class[presence].name := nom ;
                class[presence].bi := bi ;
                class[presence].bs := bs ;
                class[presence].zone := zone ;
                class[presence].effectif := effectif ;
                for i:=bi to bs do ref[zone,i] := presence ;

                nb_clas := nb_clas + 1 ;

             end;
end;
```

```
[global]procedure kill_clas(nbc : integer);

{  supprime la nbcième classe du tableau des classes
   (pas du fichier des classes !)  }

var     i : integer ;

begin
        if not (nbc in [1..nb_clas] )
        then writeln('erreur in kill_class;
                                        unable to kill class ',nbc:1)
        else begin

                for i:=class[nbc].bi to class[nbc].bs
                do ref[class[nbc].zone,i] := 0 ;

                for i:= nbc + 1 to nb_clas
                do class[i-1]:=class[i] ;

                nb_clas := nb_clas - 1 ;

             end;
end;



[global]procedure print_clas;

{  imprime la liste des classes et leurs caractéristiques  }

var     i:integer;

begin
writeln
('N° CRITERE  NOM DE LA CLASSE   EFFECTIF CODE : DE ... A ...');
writeln
('=============================================================');
        for i:=1 to nb_clas
        do begin
                write  (i:3,class[i].zone:7,class[I].name:30);
                write(class[i].effectif:16);
                writeln(class[i].bi:7,class[i].bs:7);
           end;
end;
```

```
[global]procedure  reduce(var frequency : big_matrix ;
                                    zone1,zone2 : integer );

{  supprime les lignes et les colonnes nulles de frequency
  répercute la réduction sur les classes mais pas dans le fichier  }
{  autrement dit supprime les classes d'effectif nul  }

var     i,j     : integer ;
                  {  indice ligne et colonne respectivement  }

        b       : big_matrix ; {  matrice de travail  }

        dv,dh   : integer ;{  decalage vertical et horizontal  }

        killed_clas     : integer ;
                          {  nombre de classes supprimees      }

begin
        b := frequency ;

{
 Suppression des classes sans code : passage à une sous-population
{  Ceci s'effectue pour les critères 8 et 9  }

        for i:=1 to nb_clas
        do if ((class[i].zone in [zone1,zone2])
           and (class[i].zone in [8,9])
           and (index(class[i].name,'Sans') in [1,2]))
           then class[i].effectif := 0 ;

{  suppression des lignes de zeros avec decalage vers le haut  }

        dv := 0;
        for i:=1 to nb_clas
        do if ((class[i].effectif = 0)
           and (class[i].zone in [zone1,zone2]))
           then dv := dv + 1
           else for j:=1 to nb_clas do b[i -  dv,j]:=b[i,j] ;

{  suppression des colonnes de zeros avec decalage vers la gauche  }

        dh := 0;
        for j:=1 to nb_clas

        do      if ((class[j].effectif = 0)
                and (class[j].zone in [zone1,zone2]))
                then dh := dh + 1
                else for i:=1 to nb_clas do b[i,j - dh]:=b[i,j];

        frequency := b;
```

```
{  supprimer les classes vides (dont l'effectif est nul)  }

        killed_clas := 0 ;
        j := 1 ;
        while j<= nb_clas

       do if ((class[j].effectif = 0) and (class[j].zone in [zone1,z
           then begin

                writeln('classe[',j:2,'] (',class[j].name:30,') vide'
                 killed_clas := killed_clas + 1 ;
                 kill_clas(j) ;
                 end
           else j:= j + 1;

{
 le nombre de classes supprimees doit etre egal au nombre de colonne
   et de lignes supprimees  }

        writeln ;
        if killed_clas > 0
        then writeln(killed_clas:1,' classes ont été supprimées');

        if not (dh = dv)

       then writeln(chr(7),'[From module Build] : Error during reduc
end;
```

```
[global]procedure build_clas(var frequency : big_matrix ;
                                        zone1,zone2 : integer);

{  calcule les effectifs resultant de l'intersection
   de la zone1 et de la zone2  }

var     f1,f2    : text ;

        line_1 , line_2 : varying[30] of char ;

        i,j      :integer;
                        {  compteurs a usages multiples          }

        nb1      : integer ; {  NomBre de Lignes lues  }

        nb_code : integer ;

        code1,code2      : array[1..10] of integer;

        nb_code1,nb_code2: integer ;
                            {  taille des tableaux code1 _code2  }

        fn1,fn2 : varying[40] of char ;
             {  nom des fichiers contenant les ocdes des 2 zones  }

begin
        writev(fn1,'c1',zone1:1,'.res');
        writev(fn2,'c1',zone2:1,'.res');
        open (f1,file_name:=fn1,history:=readonly);
        open (f2,file_name:=fn2,history:=readonly);
        reset(f1) ;
        reset(f2) ;

        for i:=1 to nb_clas
        do if (class[i].zone in [zone1,zone2])
           then class[i].effectif := 0 ;

        nb_code := 0 ;
        nb1 := 0 ;
        while not ( eof(f1) or eof(f2) )
        do begin

                nb1 := nb1 + 1 ;

                if zone1 in [1,2,3,4,5]
                then begin
                        readln(f1,code1[1]) ;
                        nb_code1 := 1 ;
                     end
                else begin
                        readln(f1,line_1);
                        for i:=1 to length(line_1) div 3

                          do readv(substr(line_1,(i - 1)  3 + 1,3 ),co
                           nb_code1 := length(line_1) div 3 ;
                     end ;

                if zone2 in [1,2,3,4,5]
                then begin
```

```
                    readln(f2,code2[1]) ;
                    nb_code2 := 1 ;
                end
            else begin
                    readln(f2,line_2);
                    for i:=1 to length(line_2) div 3

                   do readv(substr(line_2,(i - 1)  3 + 1,3 ),co
                    nb_code2 := length(line_2) div 3 ;
                end ;

            for i:=1 to nb_code1
            do for j:=1 to nb_code2
            do
            begin

            nb_code := nb_code + 1 ;

            if ref[zone1,code1[i]] = 0

           then writeln('La parimie n°',nb1:1,'[code : ',code1[i

                        ' ne peut être classée selon le critère'
            if ref[zone2,code2[j]] = 0

           then writeln('La parimie n°',nb1:1,'[code : ',code2[j

                        ' ne peut être classée selon le critère'


           if ((ref[zone1,code1[i]] <> 0) and (ref[zone2,code2[j
            then
            begin

           frequency[ref[zone1,code1[i]],ref[zone2,code2[j]]] :=

           frequency[ref[zone1,code1[i]],ref[zone2,code2[j]]] +

           frequency[ref[zone2,code2[j]],ref[zone1,code1[i]]] :=

           frequency[ref[zone2,code2[j]],ref[zone1,code1[i]]] +
            class[ref[zone1,code1[i]]].effectif :=
            class[ref[zone1,code1[i]]].effectif + 1 ;
            class[ref[zone2,code2[j]]].effectif :=
            class[ref[zone2,code2[j]]].effectif + 1 ;
            end; {  of if  }
            end; {  of double for  }
      end; {  of while  }

    writeln(nb1:1,' lignes de code(s) ont été traitées ');
    write('Fichiers de tailles ');
    if not (eof(f1) and eof(f2)) then writeln ('différentes')
                                 else writeln('identiques');
    writeln('Effectif de la population classée :',nb_code:1) ;

    close(f1);close(f2);
end;

end.
```

```
[environment('matrix.env')]
module matop(input,output,m_file);

const   max_size = 50 ;
        min_elem = 0.0000000001 ;

type    elem_type = real;
        mat = array[1..max_size,1..max_size]  of elem_type ;
        mat_nam = varying[132] of char  ;
        matrix = record
                        nam : mat_nam              ;
                        data : mat                 ;
                        nbcol : 0..max_size        ;
                        nbline : 0..max_size       ;
                 end;

var     m_file : text; {  matrix_file  }

procedure error(nberr:integer);

{  le premier chiffre du numéro de l'erreur est le numéro
   de la procédure / fonction ayant détecté une anomalie .
   Le deuxième chiffre est le numéro de l'anomalie détectée.

 1:pas assez d'espace disque pour écrire la matrice  [save_matrix]

11:taille de la matrice trop grande (trop de lignes) [create_matrix]
12:taille de la matrice trop grande (trop de lignes) [create_matrix]

21:fichier inexistant [read_matrix]
22:taille de la matrice trop grande (trop de colonnes)[read_matrix]
23:taille de la matrice trop grande (trop de lignes  )[read_matrix]
24:mauvais format du fichier [read_matrix]

31:tailles des matrices incompatibles pour le produit matriciel [pm]

41:tailles incompatibles pour l'addition matricielle  [sum]

51:tentative de créer une matrice diagonale
     de dimension supérieure à max_size [diag]

61:tentative d'accéder à une ligne dont le numéro est supérieur
     à la taille maximum de n'importe quelle matrice [line]
62:tentative d'accéder à une ligne existante virtuellement
   mais dont le n° est inférieur à la taille de la matrice [line]

71:tentative d'accéder à une colonne dont le numéro est supérieur
     à la taille maximum de n'importe quelle matrice [col]
72:tentative d'accéder à une colonne existante virtuellement
   mais dont le n° est inférieur à la taille de la matrice [col]

81:pas possible d'étendre la matrice car celle-ci
     est déjà à sa taille maximum [totalize]

91:tentative de supprimer une colonne
   virtuellement inexistante [kill_col]

101:tentative de supprimer une ligne
    virtuellement inexistante [kill_line]
```

```
111: ligne ou colonne nulle rend impossible l'analyse
     des profils lignes      [cp1]

121: ligne ou colonne nulle end impossible l'analyse
     des profils colonnes    [cp2]

131: la matrice doit être une matrice-ligne ou colonne [norme]  }

const   fatal_error = [22,23,31,41,61,71,81,111,121,131];

{  par défaut une error n'est pas fatale  }
{  le messages d'erreurs sont en anglais pour
   augmenter la brièveté du message  }

begin
        writeln;
        write(chr(7));

        if nberr in fatal_error then  write('FATAL ERROR : ')
                                 else  write('WARNING : ');

        if nberr = 1
        then begin
                writeln('Disk quota exceeded');

               writeln('Login on another terminal, make some space')
                writeln('Type <return> when it''s done');
                readln;
             end;

        case nberr of
        31:writeln('[From Pm] : incompatible matrix size ');
        42:writeln('[From Sum] : incompatible matrix size ');
        111:writeln('From cp1] : null line or column ');
        121:writeln('From cp2] : null line or column ');

       131:writeln('[From Norme] : matrix must be one line or one co

       4:writeln('[From Line] : trying to access to inexistent line

       5:writeln('[From Column] : trying to access to inexistent col
        end;
        writeln ('error [',nberr:2,']');;
end;
```

```
[global]procedure print_matrix(a:matrix;ent,nbdec:integer);

{  imprime la matrice a à l'écran.
   ent    : nombre maximum de caractères écrits par nombre ttc
   nbdec : nombre de decimales par nombre

   Erreur : Si une ligne de chiffres dépasse 120 caractères,

            Elle sera imprimée en plusieurs lignes de 120 caractères

var     i,j     : integer;
        skip    : integer ; {  nombre de nombres par ligne  }
        ligne   : varying[200] of char ;
begin

with a do begin
                skip:=120 div ent ;
                writeln(a.nam,' [',nbline:2,' X ',nbcol:2,' ]');
                for i:=1 to length(nam)+11 do write('=');
                writeln;
                for i:=1 to nbline do
                        begin
                        ligne := '';
                        for j:=1 to nbcol
                        do begin
                                write(data[i,j]:ent:nbdec);
                                if j mod skip = 0
                                then writeln;
                           end;
                        writeln;
                        end;
        end;
end;
```

```
[global]procedure save_matrix(a:matrix;name:mat_nam);

{  sauve la matrice a sur un fichier de type text
   (lisible et modifiable grâce à un éditeur)
   Name sera le nom du fichier (suffixe ".dat" par défaut)

   Erreur : S'il n'y a plus suffisamment d'espace disque
            pour sauver le fichier : l'utilisateur sera
            invité à faire de la place sur disque (purge1)
            Cette solution permet d'éviter une interruption
            définitive du programme  }

var     i,j:integer;
        floating : boolean ; {  true ssi la matrice est réelle  }

begin
with a do begin
                floating := false ;
                for i:=1 to nbline
                do for j:=1 to nbcol
                do if abs(round(data[i,j]) -  data[i,j]) > min_elem
                   then floating := true ;

                open(m_file,file_name:=name,history:=new);
                rewrite(m_file);
                writeln(m_file,nbline,nbcol,error:=continue);
                for i:=1 to nbline
                do for j:=1 to nbcol
                   do  if floating

                       then writeln(m_file,data[i,j]:32:30,error:=con

                       else writeln(m_file,data[i,j]:5 :0 ,error:=con
                if status(m_file) > 0
                then begin error(1) ;
                           readln ; save_matrix(a,name); end;
           end;
           close(m_file,save);
end;
[global]procedure create_matrix (var a:matrix;
                                  n,m:integer;mat_n:mat_nam);
{
 cree une matrice de dimension n x m avec tous les elements egaux a
{  le nom de la matrice est mat_n
   Erreur : Si n,le nombre de lignes de a, est supérieur à max_size

            Si m,le nombre de colonnes de a, est supérieur à max_size

var     i,j:integer;

begin
        if not (n in [1..max_size]) then error(11);
        if not (m in [1..max_size]) then error(12);
        for i:=1 to max_size do
        for j:=1 to max_size do a.data[i,j]:=0;
        a.nbcol  := m;
        a.nbline := n;
        a.nam    := mat_n;
end;
```

```
[global]procedure read_matrix(var a : matrix;name :mat_nam);

{  lit une matrice se trouvant sur un fichier de type text ;
   name est le nom du fichier contenant la matrice
   Erreur : Si la matrice porte le nom d'un fichier inexistant
            Si le nombre de lignes de a est supérieur à max_size

          Si le nombre de colonnes de a est supérieur à max_size

var     i,j:integer;

begin
with A do
begin

      open(m_file,file_name:=name,history:=readonly,error:=continue
       if status(m_file) > 1
       then error(21)
       else begin
              reset(m_file);
              readln(m_file,nbline,nbcol);
              if not (nbline in [1..max_size]) then error(22);
              if not (nbcol  in [1..max_size]) then error(23);
              create_matrix(a,nbline,nbcol,name);
              for i:=1 to nbline do
                     for j:=1 to nbcol
                     do begin
                            read(m_file,data[i,j]);
                            if eoln(m_file)
                            then readln(m_file,error:=continue);
                            if status(m_file) > 1
                            then error(24);
                        end;
              close(m_file);
            end;
       end;
end;
```

```
[global]function pm(a,b: matrix ) : matrix ;
{  calcule le Produit Matriciel A X B
   Erreur : Si les tailles de A et B sont incompatibles
   c-à-d si le nombre de colonnes de A est différent
   du nombre de colonnes de B  }

var     i,j,k : integer ;
        c       : matrix;
begin
        if a.nbcol<>b.nbline then error(31)
        else begin
                c.nbline:=a.nbline;
                c.nbcol := b.nbcol;
                for i:=1 to a.nbline do
                for j:=1 to b.nbcol
                do begin
                   c.data[i,j]:=0;
                   for k:=1 to a.nbcol do
                   c.data[i,j]:=c.data[i,j]+a.data[i,k]b.data[k,j]
                   end;
             end;
        c.nam:=a.nam + ' X ' + b.nam;
        pm:=c;
end;

[global]function sum(a,b:matrix):matrix;

{  calcule la somme de deux matrice de même taille
   Erreur : Si les tailles sont différentes  }

var     i,j     :integer;
        c       :matrix ;
begin
        if ((a.nbline<>b.nbline) or (a.nbcol<>b.nbcol))
        then error(41)
        else begin
                c:=a;
                for i:=1 to a.nbline do for j:=1 to a.nbcol
                do c.data[i,j]:=c.data[i,j]+b.data[i,j];
             end;
        sum:=c;
end;
[global]function ps(c:elem_type;a:matrix):matrix;

{  calcule le produit scalaire de ca  }

var     i,j : integer ;

begin
        for i:=1 to a.nbline do
        for j:=1 to a.nbcol
        do a.data[i,j]:=a.data[i,j]c;
        writev(a.nam,c:2:2,'',a.nam);
        ps:=a;
end;
```

```
[global]function diag(n:integer):matrix;

{  crée une matrice unite de dimension n x n
   Erreur : si n<=0 ou n>max_size                }

var     res     : matrix;
        i,j     : integer;


begin
        if not (n in [1..max_size]) then error(51);
        res.nbcol  := n ;
        res.nbline := n ;
        for i:=1 to n do for j:=1 to n
        do if i=j then res.data[i,j] := 1
                  else res.data[i,j] := 0;
        writev(res.nam,'1(',n:1,')');
        diag:=res;
end;
[global]function trans(a:matrix):matrix;
{  calcule la transposee de la matrice A  }

var     i,j : integer;
        b:matrix;
begin
        for i:=1 to a.nbline
        do for j:=1 to a.nbcol do b.data[j,i]:=a.data[i,j];
        b.nbline:=a.nbcol;
        b.nbcol :=a.nbline;
        b.nam:='(' + a.nam + ')'';
        trans := b;
end;
function col(a:matrix ; n:integer):matrix;

{
  col sera une matrice a.nbline x 1 contenant la nieme colonne de a
   Erreur : ssi n est le numéro d'une colonne inexistante de a
   soit n > nombre de colonnes de a ou n<=0 : erreur fatale

       n > nombre de colonnes de a et n<= max_size : avertissement

var     i:integer;
        b:matrix;

begin
        if n in [a.nbline+1..max_size] then error(61);
        if not(n in [1..max_size]) then error(62);
        for i:=1 to a.nbline do b.data[i,1]:=a.data[i,n];
        b.nbline := a.nbline;
        b.nbcol  := 1 ;
        writev(b.nam,'col[',n:1,'] de ',b.nam);
        col := b;
end;
```

```
function line(a:matrix ; n:integer):matrix;

{  line sera une matrice 1 x a.nbcol contenant la nieme ligne de a
   Erreur : ssi n est le numéro d'une colonne inexistante de a
   soit n > nombre de lignes de a ou n<=0 : erreur fatale

        n > nombre de lignes de a et n<= max_size : avertissement   }

var     j:integer;
        b:matrix ;
begin
        if (n in [a.nbcol+1..max_size]) then error(71);
        if not(n in [1..max_size]) then error(72);
        for j:=1 to a.nbcol do b.data[1,j]:=a.data[n,j];
        b.nbline := 1;
        b.nbcol  := a.nbcol;
        writev(b.nam,'ligne[',n:1,'] de ',b.nam);
        line := b;
end;
[global]procedure totalize (var k:matrix);
{  ajoute a la matrice k une ligne et une colonne de totaux  }

var     i,j:integer;

begin
with k do
begin
        if ((nbline >= max_size) or (nbcol>= max_size))
        then error(81) ;
        for i:=1 to nbline+1 do data[i,nbcol+1]:=0;
        for j:=1 to nbcol do data[nbline+1,j]:=0;
        for i:=1 to nbline do
        for j:=1 to nbcol
        do begin
                data[nbline+1,j]:=data[nbline+1,j] + data[i,j];
                data[i,nbcol+1] := data[i,nbcol+1] + data[i,j];
          end;
        for i:=1 to nbline

       do data[nbline+1,nbcol+1]:=data[nbline+1,nbcol+1]+data[i,nbco
        nbcol := nbcol + 1 ;
        nbline := nbline + 1 ;
end;
end;
```

```
[global]procedure kill_zeroes(var a :matrix);

{  supprime les lignes et les colonnes de zeros d'une matrice a
   en la compactant  }

var     i,j     : integer ;
                  {  indices ligne et colonne respectivement  }
        b       : matrix  ;{  matrice de travail  }
        dv,dh   : integer ;{  decalage vertical et horizontal  }

begin
        b := a ;
        totalize(b) ;

{  suppression des lignes de zeros avec decalage vers le haut  }

        dv := 0;
        for i:=1 to b.nbline
        do      if b.data[i,b.nbcol + 1] = 0
                then dv := dv + 1

               else for j:=1 to b.nbcol do b.data[i -  dv,j]:=b.data
        b.nbline := b.nbline - dv;

{  suppression des colonnes de zeros avec decalage vers la gauche  }

        dh := 0;
        for j:=1 to b.nbcol
        do      if b.data[b.nbline + 1,j] = 0
                then dh := dh + 1

               else for i:=1 to b.nbline do b.data[i,j - dh]:=b.data
        b.nbcol := b.nbcol - dh;

        a := b;
end;

[global] procedure kill_col(var a:matrix;n:integer);

{  supprime la ligne n de la matrice a
   message d'erreur : ssi la ligne n de a n'existe pas  }

var     i,j     : integer ;

begin
        if not (n in [1..a.nbcol])
        then error(91)
        else
        begin
                for j:=n+1 to a.nbcol

               do      for i:=1 to a.nbline do a.data[i,j -1] :=a.da
                a.nbcol := a.nbcol - 1 ;
        end;
end;
```

```
[global] procedure kill_line(var a:matrix;n:integer);

{  supprime la colonne n de la matrice a
   message d'erreur : ssi la colonne n de a n'existe pas  }

var     i,j     : integer ;

begin
        if not (n in [1..a.nbline])
        then error(101)
        else
        begin
                for i:=n+1 to a.nbline

               do       for j:=1 to a.nbcol do a.data[i-1,j] :=a.data
                a.nbline := a.nbline - 1 ;
        end;
end;

procedure fvp;fortran;

[global]procedure class_vp(var l,u : matrix);
{
 classe les valeurs propres et les vecteurs propres par ordre croiss
   des valeurs propres ;
                les vecteurs propres sont les colonnes de u       }

procedure intervert(var a,b : elem_type) ;
{  intervertit les valeurs de a et b  }
var     inter   : elem_type ;
begin inter := a ; a:=b ; b:=inter ; end ;

var     x       : matrix  ;
        i,j     : integer ;
        ordre   : array [1..max_size] of elem_type ;
begin

        for i:=1 to max_size do ordre[i] := i ;

        for i:=1 to l.nbline
        do for j:=i+1 to l.nbline do
          if l.data[j-1,1] < l.data[j,1]
          then begin
                intervert(l.data[j-1,1],l.data[j,1]);
                intervert(ordre[j-1],ordre[j]);
               end;

{  a ce moment les valeurs propres sont triees et ordre[i] contient

  le numéro du vecteur propre (dans u) associé à la ième valeur prop

        x := u ;

        for i:=1 to u.nbcol
        do begin
                for j := 1 to u.nbline
                do u.data[j,i] := x.data[j,round(ordre[i])];
           end;
end;
```

```
[global]procedure vp(a:matrix;var vap,vep : matrix);
{  calcule les vecteurs et valeurs propres de la matrice A  }

var     i,j     :integer ;

begin
        save_matrix(a,'FOR001.DAT');
        fvp;  {  les V.P. sont calculees par routines fortran ;
                 le passage des parametres se fait par fichiers  }
        read_matrix(vap,'FOR002.DAT');
                                        vap.nam:='val. pr. de ' + a.nam;
        read_matrix(vep,'FOR003.DAT');
                                        vep.nam:='vec. pr. de ' + a.nam;
end;
[global]procedure verify (a,l,x:matrix);
{  verifie si l[i]x[i] = ax[i] ,autrement dit

          si l et x sont respectivement valeurs et vecteurs propres

var     i:integer;

begin
        for i:=1 to l.nbline do
        print_matrix
        (trans(sum(pm(ps(-1.data[i,1],diag(a.nbcol)),
               col(x,i)),pm(a,col(x,i)))),9,6);

end;
function norme(x,m:matrix):elem_type;
{  Q est la matrice du produit scalaire  }
{  la norme de x vaut donc le radical de x2 = x'q x }
{  cfr bourtier page 66 (ici, m = q )  }

var     b:matrix;

begin
        if not((x.nbline=1) or (x.nbcol=1))
        then error(131);
        b:=pm(trans(x),pm(m,x));
        norme := sqrt(b.data[1,1]);
end;
```

```
[global]procedure cpl(k:matrix;var z,lambda : matrix);

var     f,m,v,w,u        : matrix ;
        di,dj,qi,qj,x    : matrix ;
        pi,pj,c,a,s,d    : matrix ;
        i,j,l:integer ;
        n,p:integer;
        vt,inertie : elem_type ;
        nbind    : integer ;

begin
        n:=k.nbline;
        p:=k.nbcol;

{  calcul de x et y page 129 _130  }

        x:=k;
        totalize(k);

        nbind := trunc(k.data[n+1,p+1]);
writeln('analyse
        des ',x.nbline:1,' profils lignes (',nbind:1,' individus)');

        for i:=1 to n
        do for j:=1 to p
        do x.data[i,j]:=x.data[i,j]/(k.data[i,p+1]);

{  calcul des poids  }

        create_matrix(qi,p,p,'qi');
        create_matrix(pi,n,n,'poids des lignes');
        create_matrix(qj,n,n,'qj');
        create_matrix(pj,p,p,'poids des colonnes');
        for i:=1 to n do pi.data[i,i]:=k.data[i,p+1]/nbind;
        for i:=1 to n do qj.data[i,i]:= 1/pi.data[i,i];
        for j:=1 to p do pj.data[j,j]:=k.data[n+1,j]/nbind;
        for j:=1 to p do qi.data[j,j]:= 1/pj.data[j,j];

        d := pi;
        m := qi;
        v:=pm(trans(x),pm(d,x));

        s := pm(v,m) ;

        vp(s,lambda,u);
        class_vp(lambda,u);

        for j:=1 to u.nbcol do begin
                                vt:=norme(col(u,j),m);
                                if vt = 0 then error(111);
                                for i:=1 to u.nbline
                                do u.data[i,j]:=u.data[i,j]/vt;
                               end;
        a:=pm(m,u);
        z := a ;
        z.nam := 'projection des points';

{  calcul de l'inertie et de la part d'inertie expliquee  }
```

```
        inertie := 0;

       for j:=2 to lambda.nbline do lambda.data[j-1,1]:=lambda.data[
        lambda.nbline := lambda.nbline - 1 ;

       for i:=1 to lambda.nbline do inertie := inertie + lambda.data

       for i:=1 to lambda.nbline do lambda.data[i,2]:=lambda.data[i,
        lambda.data[1,3]:=lambda.data[1,2];
        for i:=2 to lambda.nbline
        do lambda.data[i,3]:=lambda.data[i-1,3] + lambda.data[i,2];
        lambda.nbcol := 3;
        lambda.nam :=

        '       valeur propre     part d''inertie expl.  inertie tot.
end;

[global] procedure cp2(k:matrix;var z,lambda : matrix);

var     f,x,g,m,v,w,u    : matrix ;
        di,dj,qi,qj,y    : matrix ;
        pi,pj,c,a,s,d    : matrix ;
        vep              : matrix ;
        i,j,l:integer ;
        n,p:integer;
        vt,inertie : elem_type ;
        nbind    : integer ;

begin
        n:=k.nbline;
        p:=k.nbcol;

{  calcul de x et y page 129 _130  }

        y:=k;
        totalize(k);
        nbind := trunc(k.data[n+1,p+1]);

writeln('analyse
        des ',y.nbline:1,' profils lignes (',nbind:1,' individus)');

        for i:=1 to n
        do for j:=1 to p
        do y.data[i,j]:=y.data[i,j]/k.data[n+1,j];

{  calcul des poids  }

        create_matrix(qi,p,p,'qi');
        create_matrix(pi,n,n,'poids des lignes');
        create_matrix(qj,n,n,'qj');
        create_matrix(pj,p,p,'poids des colonnes');
        for i:=1 to n do pi.data[i,i]:=k.data[i,p+1]/nbind;
        for i:=1 to n do qj.data[i,i]:= 1/pi.data[i,i];
        for j:=1 to p do pj.data[j,j]:=k.data[n+1,j]/nbind;
        for j:=1 to p do qi.data[j,j]:= 1/pj.data[j,j];

        d := pj;
        m := qj;
        v:=pm(pm(y,d),trans(y));
```

```
        s := pm(v,m) ;

        vp(s,lambda,u);
        class_vp(lambda,u);

        for j:=1 to u.nbcol do begin
                                  vt:=norme(col(u,j),m);
                                  if vt = 0 then error(121);
                                  for i:=1 to u.nbline
                                  do u.data[i,j]:=u.data[i,j]/vt;
                                end;
        a:=pm(m,u);

        z:=a;
        z.nam := 'projection des points';

{  calcul de l'inertie et de la part d'inertie expliquee  }

        inertie := 0;

       for j:=2 to lambda.nbline do lambda.data[j-1,1]:=lambda.data[
        lambda.nbline := lambda.nbline - 1 ;

       for i:=1 to lambda.nbline do inertie := inertie + lambda.data

       for i:=1 to lambda.nbline do lambda.data[i,2]:=lambda.data[i,
        lambda.data[1,3]:=lambda.data[1,2];
        for i:=2 to lambda.nbline
        do lambda.data[i,3]:=lambda.data[i-1,3] + lambda.data[i,2];
        lambda.nbcol := 3;
        lambda.nam :=

        '       valeur propre     part d''inertie expl.  inertie tot.

end;

end.{  of module matrix  }
```

```
[environment('string.env')]

module string (input,output,pf) ;

const   maj = ['A','B','C','D','E','F','G','H','I','J','K','L','M',
              'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'];

        min = ['a','b','c','d','e','f','g','h','i','j','k','l','m',
              'n','o','p','q','r','s','t','u','v','w','x','y','z'];

        digit = ['1','2','3','4','5','6','7','8','9','0'] ;


       accent = ['"','','~','|','''','_','.',',',':','!','?','-'] ;

        aac = ['à','á','â','ã','ä'];
        eac = ['è','é','ê','ë'];
        iac = ['ì','í','î','ï'];
        oac = ['ð','ó','ô','õ','ö'];
        uac = ['ù','ú','û','ü'];
        nac = ['ñ','ñ'];

        voyelle = ['a','e','i','o','u'] ;
        cons    = ['b','c','d','f','g','h','j','k','l','m',
                   'n','p','q','r','s','t','v','w','x','y','z'];

type    phrase = varying[200] of char ;
        name    = varying[80] of char;
        mot = varying[40] of char ;

var     pf       : text ;
        w        : array[1..40] of mot ;
        nbw      : integer ;
        verbosity : boolean ;

[global]function equal(a,b:mot):boolean ;

begin
        if length(a) <> length(b) then equal := false
                                  else equal := (a=b);
end;
```

```
[global]procedure desaccentuer(a:phrase ; var b : phrase);

{
      a : phrase en rundi a traiter ( les 10 premieres lettres sont
                                      un identifiant non pris en ligne de c
          et la phrase ne contient pas deux blancs consecutifs ;

      b : contiendra la phrase en rundi sans les accents
          b ::= <' '><mot> ... <' '><mot><' '>
          <mot> ::= <suite non-vide de lettres minuscules>

   en outre convertit toute majuscule en minuscule
   quatres sortes de caracteres peuvent se presenter :
        un accent "pur"
        un backspace
        une lettre sans accent
        une lettre accentuee    }

var     i       : integer ;
        r       : real ;
        ch      : char ;

begin
                b := '' ;
                readv(a,r,a);
                for i:=1 to length(a)
                do begin
                        ch := a[i] ;
                        if ch = '''' then b := b + ''' '
                   else if ch in min then b := b + ch
                   else if ch = ' '  then begin
                                          if length(b) > 0
                                          then if b[length(b)]<>' '
                                               then b := b + ' ' ;
                                          end
                   else if ch in maj then b := b + chr(ord(ch)+32)
                   else if ch in aac then b := b + 'a'
                   else if ch in eac then b := b + 'e'
                   else if ch in iac then b := b + 'i'
                   else if ch in oac then b := b + 'o'
                   else if ch in uac then b := b + 'u'
                   else if ch in nac then b := b + 'n'
                   end;
                if b[1]<>' ' then b := ' ' + b ;
                if b[length(b)]<>' ' then b:= b + ' ' ;
end;
```

```
function revert(a : mot) : phrase ;

{  renverse l'ordre des lettres de la phrase a  }

var     i : integer ;
        b : mot     ;

begin
        b := '';
        for i := length(a) downto 1
        do b := b + a[i];
        revert := b ;
end;

procedure dcm(a:phrase) ;
{  decompose la phrase a en mots
   w sera le tableau des mots et nbw sa longueur
   w et nbw sont des variables globales  }

var     i       : integer ;

begin
        nbw := 0 ;
        if ((pad ('',' ',length(a)) <> a) and (length(a) > 0))

       then  {  la phrase contient au moins un caractère nonblanc  }
        begin
        while a[1] = ' ' do a := substr(a,2,length(a) - 1);
        while a[length(a)] = ' ' do a := substr(a,1,length(a) - 1 );
        a := ' ' + a + ' ' ;
        while index(a,'  ') <> 0
        do begin
                i := index(a,'  ') ;

              a := substr(a,1,i-1) + substr(a,i+1,length(a) - i ) ;
           end ;

        for i := 1 to length(a)-1
        do begin
                if a[i] = ' ' then
                                  begin
                                  nbw := nbw + 1;
                                  w[nbw] := '';
                                  end
                             else w[nbw] := w[nbw] + a[i] ;
           end
        end;
end;
```

```
[global]function choice (l:name):char;

{  lit un caractère au terminal jusqu'au moment où celui-ci
   est une des lettres contenues dans l  }

var     i       :integer;
        pos     : set of char;
        ch      :char;
begin
        pos := [];
        for i:=1 to length(l) do pos:= pos + [l[i]];
        repeat  begin
                readln(ch);
                if not (ch in pos) then write (chr(7))
                                   else choice:=ch;
                end
        until ch in pos;
end;


[global] procedure capitalize (var a : phrase );

{  transforme les lettres de a en lettres majuscules  }

var     i : integer ;

begin
        for i := 1 to length(a)
        do begin
                if a[i] in aac then a[i] := 'a';
                if a[i] in eac then a[i] := 'e';
                if a[i] in iac then a[i] := 'i';
                if a[i] in oac then a[i] := 'o';
                if a[i] in uac then a[i] := 'u';
                if a[i] in min then a[i] := chr(ord(a[i]) - 32) ;
           end;
end;

function fill( a : phrase ;
                           app : phrase ;lg : integer) : phrase ;

{  semblable à pad mais ne produit pas d'erreur

  si la longueur du string est supérieure à la longueur demandée (lg

begin
        while length(a) + length(app) <= lg
        do a := a + app ;
        fill := a ;
end;
```

```
function center(lign: phrase ; longueur : integer) : phrase ;

{  met autant de blanc au début qu'à la fin de la phrase  }
{  autrement dit centre la phrase  }

var     i,j     : integer ;
        nbb     : integer ;
        corps   : phrase  ;

begin
        lign := fill (lign,' ',longueur);

        if pad('',' ',length(lign)) <> lign
{  il existe au moins un caractère non blanc dans ligne  }
        then
        begin
        nbb := 0 ;
        i := 1 ;
        while lign[i] = ' ' do  begin i:= i + 1 ;
                                      nbb := nbb + 1 ; end ;
        j := length(lign) ;
        while lign[j] = ' ' do begin j:= j - 1 ;
                                      nbb := nbb + 1 ; end ;
        corps := substr(lign,i,length(lign) - nbb );
        lign := fill('',' ',nbb div 2) + corps ;
        lign := fill(lign,' ',length(corps) + nbb );
        end ;

        center := lign ;
end ;

end. {  of module string  }
```

```
[inherit('bt.env','matrix.env')]
program build(input,output);

var     zone1,zone2 : integer ;

        frequency       : big_matrix ;

       { frequency [i,j] est le montant total des paremies appartena
         simultanement a la classe[i] ET a la classe[j]          }

        i,j     : integer ;

        analcp  :matrix;
               {  sous-matrice resultant du croisement de 2 zones  }

        fn      : varying[40] of char;
                                    {  File_Name : nom de fichier  }

        n,m     :integer ;
```

```
( main )
begin
        write('creer une table de contingence entre : ');
        readln(zone1,zone2) ;

       for i:=1 to max_clas do for j:=1 to max_clas do frequency[i,j

        read_clas;      { lire la partition en classes demandee }

        build_clas(frequency,zone1,zone2);
                { effectuer la partition sur les donnees codees }

        reduce(frequency,zone1,zone2);
     { eliminer les lignes/colonnes de zeros et les classes vides }


       { construction des sous_matrices et sauvetage sur fichier }

        with analcp do
            begin

{
 recherche des coordonnees de la sous-matrice de frequency qui conti

  les effectifs des classes resultant de l'intersection entre les cr
   zone1 _zone2  }

                i := 0 ;
                repeat i := i + 1 until class[i].zone = zone1 ;
                nbline := nb_clas + 1;

               repeat nbline := nbline -  1 until class[nbline].zone

        { critère zone1 : classes [i..nbline]  }

                j := 0 ;
                repeat j:=j + 1 until class[j].zone = zone2 ;
                nbcol  := nb_clas + 1 ;

               repeat nbcol := nbcol -  1 until class[nbcol].zone =

        { critère zone2 : classes [j..nbcol]  }

                for n:=i to nbline do for m:=j to nbcol
                do analcp.data[n - i + 1,m - j + 1]:=frequency[n,m];

                nbline := nbline - i + 1 ;
                nbcol  := nbcol  - j + 1 ;

                writev(fn,'FOR0',zone1:1,zone2:1,'.dat');
                save_matrix(analcp,fn);
                writev(fn,'FOR0',zone2:1,zone1:1,'.dat');
                save_matrix(trans(analcp),fn);

                save_clas;
        end;
end.
```

```
[inherit('string.env')]

program find_symbol(input,output,res);

const   l_max    = 45 ;
        h_max    = 200;

var     a        : varying[200] of char ;

        i,j,h,l  : integer ;

        tab      : array[1..h_max,0..l_max] of boolean ;

        title    : array[1..h_max] of name ;

        list     : array[1..h_max] of phrase ;

        ok       : boolean ;

        lg       : integer ;

        res      : text ;

        suf,fn   : name ;

        zone1,zone2 : integer ;

function locate(var i, j : integer):boolean ;

var     temp     :boolean ;

begin
temp:=false;
j:=2;
while ((j<l) and (not temp))
do begin
   i:=2;
   while ((i<= h) and (not temp))
   do begin

        temp := tab[i,j-1] and tab[i,j+1] and tab[i,j] and tab[i-1,j]
                 and (not tab[i-1,j-1]) and (not tab[i-1,j+1]);
         i := i +  1 ;
      end ;
   j := j + 1;
   end;
i := i - 1 ;
j:=j - 1;
locate := temp ;
end;
```

```
function find_length(i,j : integer):integer ;

var     k:integer ;

begin
        k:=0 ;
        while tab[i,j+k] do k:=k+1;
        find_length:=k;
end;

procedure remonter (i,j,lg : integer);

var     k,m,centre,y :integer ;
        exact   : boolean ;

begin
        k := 0;
        while (tab[i-k,j]) do k := k + 1 ;

{  k = le premier blanc après la ligne verticale  }

        centre := i - ((k) div 2) ;

        if k div 2 = k / 2 then exact:=true
                           else exact:=false;


        for y:=centre to i
        do for k:=j+1 to lg+j-1 do tab[y,k] := false ;

        for k:=j+1 to lg+j-1 do tab[centre,k] := true ;

        if exact then for k:=j to lg+j-1 do list[centre,k] := '_'
                 else for k:=j to lg+j-1 do list[centre,k] := '-' ;


        if tab[i+1,j+lg-1]
        then for k:=centre to i do tab[k,lg+j-1] := true
        else for k:=centre to i-1 do tab[k,lg+j-1] := false;
end;
```

```
procedure printab ;

var     k,m     :integer ;

begin
        for k := 1 to h
        do begin for m:=1 to l
                 do  if tab[k,m] then write('') else write(' ');
                 writeln;
           end;
end;

begin
        for i:=1 to h_max do for j:=0 to l_max do tab[i,j]:=false;
        for i:=1 to h_max do list[i] := fill('',' ',l_max);


        open(pf,file_name:='for008.dat',history:=readonly);

        readln(zone1,zone2) ;
        readln(suf);
        writev(fn,'clu',zone1:1,zone2:1,'.',suf);

        l := 40 ;
        reset(pf);
        h := 1;
        while not eof(pf)
        do begin
                readln(pf,a);
                if index(a,'')<>0
                then begin
                        h := h + 1 ;
                        title[h] := substr(a,5,23);
                        a := substr(a,29,length(a)-28);
                        for i:=1 to length(a)
                        do if a[i] =  ' '  then tab[h,i] := false
                                           else tab[h,i] := true;
                     end;
           end;
        for i:=1 to l do tab[h,i]:= true ;

{  traitement des _|_  }

{  détection d'un cas  }

        ok := locate(i,j)  ;
        repeat
                begin
                lg := find_length(i,j)  ;
                remonter (i,j,lg) ;
                ok := locate(i,j) ;
                end
        until not ok ;

        close(pf) ;
        open (res,file_name := fn,history:=new);
        rewrite(res) ;

        for i := 1 to h
```

```
        do begin
                write(res,title[i]);
                for j:=1 to l
                do  if tab[i,j]
                    then
                        begin
                        if (not tab[i,j-1])
                        then list[i][j] := '|' ;

                        if ((tab[i-1,j] and tab[i,j-1] )
                        or  (tab[i+1,j] and tab[i,j-1] ))
                        then if list[i][j-1] = '|'
                             then list[i][j] := '|'
                             else if list[i-1][j-1] = '_'
                                  then list[i][j] := ' '
                                  else list[i][j] := '+' ;


                       if (tab[i,j] and not (list[i][j] in ['|','+',
                        then list[i][j] := '-';

                        if ((j in [1]) and (list[i][j] = '|'))
                        then list[i][j] := '<';
                        write(res,list[i][j]);

                        end
                    else write(res,' ');
                writeln(res);
           end;

end.
```

```
[ inherit('string.env','bt.env') ]
program class6(input,output,res);

type    tyrep = record
                nom : varying[40] of char;
                { valeur de la repetition }
                freq : integer;
                { nombre d'occurence de la repetition }
                debut : integer ;
                { indice de la lettre ou commence
                  la repetition dans original }
                fin     : integer ;
                end;

var     res     : text   ;
        { contiendra les resultats codés }
        original: phrase ;
        { ligne courante de pf avec tous les accents }
        a       : phrase ;
        { ligne courante de pf sans accent }
        c       : phrase ;
        { phrase a telle qu'on la prononce }
        d       : phrase ;
        { contenant uniquement les voyelles de c }
        rep     : array[1..20] of tyrep;
        { tableau des repetitions }
        nbr     : integer ;
        { nombre de repetitions de la ligne courante de pf }
        corr    : array[1..200] of integer ;
        { corr[i] est l'indice du correspondant
          de a[i] dans original }
        i,j,code: integer ;
        { compteurs }
        nbs , nbv : real ;
        { pondération des codes dans les classes
          Syllabes et Voyelle }
        nbl     : integer ;
        { numéro de la parémie courante }


procedure traitement_6(a:phrase);

{ ecrit les repetitions totales et partielles
  d'une paremie et leur frequence
  - une repetition totale est un mot qui se repete
  - une repetition partielle est une syllabe qui se repete }

var     i,j,k   : integer ; { compteurs          }
        target  : char    ; { caractere courant de a }
        pc      : integer ; { indice du caractere courant de a }
        suspect : integer ; { indice du caractere suspect de a }

begin
        pc := 1;
        nbr := 0;
        while (pc <= length(a) - 3 )
        do begin
                target := a[pc];
```

```
        { trouver toutes les suites commencant
          par target et qui se trouvent
          au dela de la position pc .
          Tout i tel que pc < i <= length(a)
          et a[i]=target est une position suspecte      }

               suspect := pc;
               repeat suspect := suspect + 1
               until ((substr(a,suspect,3)=substr(a,pc,3))
                   or (suspect >= length(a)-2));
               if (substr(a,suspect,3)=substr(a,pc,3))
               then begin { triple repetition reperee  }
                       nbr := nbr + 1 ;
                       rep[nbr].nom:='';
                       repeat  begin
                               if a[pc] = ' '
                               then begin
                                       nbr := nbr + 1 ;
                                       rep[nbr].nom := '';
                                    end;
                               rep[nbr].nom:=rep[nbr].nom + a[pc];
                               pc:= pc + 1;
                               suspect := suspect + 1;
                               end
                       until ((a[pc] <> a[suspect])
                               or (suspect  >= length(a)));
                    end
               else pc := pc + 1;
          end;

        i:=1;
        while i<= nbr
        do begin
               if length(rep[i].nom)<3

             then begin  { rep[i] est de 2 lettres au plus  }
                       for j:=i to nbr-1 do rep[j]:=rep[j+1];
                       nbr:=nbr - 1;
                    end
               else i:=i+1;
          end;

{  elimination des redondances  }

        for i:=1 to nbr
        do for j:=i+1 to nbr
               do if length(rep[i].nom) = length(rep[j].nom)
                  then if rep[i].nom=rep[j].nom
                       then begin
                           { rep[i] et rep[j] sont redondantes  }
                            for k:=j to nbr-1 do rep[k]:=rep[k+1];
                            nbr:=nbr - 1;
                            end;

end;


procedure alliteration(a:phrase);
```

```
{  détecte les allitérations  }

var     i,j,k    : integer ; {  compteurs          }
        target   : char    ; {  caractere courant de a  }
        pc       : integer ; {  indice du caractere courant de a  }
        suspect  : integer ; {  indice du caractere suspect de a  }
        deb      : integer ;
                        {  début de la 2ème valeur de la répetition }
begin
        pc := 1;
        nbr := 0;
        while (pc <= length(a) - 3 )
        do begin
                target := a[pc];
                suspect := pc;
                repeat suspect := suspect + 1
                until ((substr(a,suspect,2)=substr(a,pc,2))
                    or (suspect >= length(a)-1));
                if (substr(a,suspect,2)=substr(a,pc,2))
                then begin {  double répétition repérée  }
                        deb := suspect ;
                        nbr := nbr + 1 ;
                        rep[nbr].nom:='';
                        repeat  begin
                                if a[pc] = ' '
                                then begin
                                        nbr := nbr + 1 ;
                                        rep[nbr].nom := '';
                                     end;
                                rep[nbr].nom:=rep[nbr].nom + a[pc];
                                pc:= pc + 1;
                                suspect := suspect + 1;
                                end
                        until ((a[pc] <> a[suspect])
                                or (pc + 1 >= deb )
                                or (suspect  >= length(a)));
                     end
                else pc := pc + 1;
           end;

        i:=1;
        while i<= nbr
        do begin
                if length(rep[i].nom)<= 1

               then begin  {  rep[i] est de 2 lettres au plus  }
                        for j:=i to nbr-1 do rep[j]:=rep[j+1];
                        nbr:=nbr - 1;
                     end
                else i:=i+1;
           end;

{  elimination des redondances  }

        for i:=1 to nbr
        do for j:=i+1 to nbr
                do if length(rep[i].nom) = length(rep[j].nom)
                   then if rep[i].nom=rep[j].nom
                        then begin
```

```
                                { rep[i] et rep[j] sont redondantes }
                                 for k:=j to nbr-1 do rep[k]:=rep[k+1];
                                 nbr:=nbr - 1;
                                 end;

end;

procedure ecrire_6(a:phrase);

var     i,j,k : integer;
        trait,b : phrase ;
begin
        trait := '-=+~ °;
        b:='';
        b:=pad(b,' ',79);
        for i:=1 to nbr
        do begin
                rep[i].freq := 0 ;
                for j:=1 to length(a)-length(rep[i].nom)
                do if rep[i].nom = substr(a,j,length(rep[i].nom))
                   then begin
                        for k:=j to j+length(rep[i].nom) - 1
                        do b[k] := trait[i] ;
                        rep[i].freq:=rep[i].freq + 1 ;
                        end;
           end;
        if ((nbr > 0 ) and verbosity)
        then begin
                writeln;
                writeln(original);
                writeln;
                writeln('           ',a);
                writeln('           ',b);
             end;

        if verbosity then
        for i:=1 to nbr
        do begin
                for j:=1 to 8 do write(trait[i]);
                write('>>> ');
                write('Repetition ');
                if ((rep[i].nom[1]=' ')
                and (rep[i].nom[length(rep[i].nom)]=' '))
                then write ('totale')
                else write ('partielle');
                writeln (' : [',rep[i].nom,'] (',
                                rep[i].freq:2,' X )');
                if i=nbr then   writeln;
           end;
end;
```

```
procedure build_code ( phase : integer );

var     i : integer ;
        temp : real ;
begin
        temp := 0 ;
        for i:=1 to nbr
        do temp := temp + (length(rep[i].nom)  rep[i].freq);
        if phase = 1
        then temp := round(100  (temp / length(c))) / nbs;
        if phase = 2
        then temp := round(100  (temp / length(d))) / nbv;

        if temp > 4
        then begin
                if phase = 1 then writeln(nbl:4,temp:8:2,c);
                if phase = 2 then writeln(nbl:4,temp:8:2,d);
                temp := 4 ;
             end ;

        if phase = 1 then code := round(temp)10 ;
        if phase = 2 then code := code + round(temp);
        if verbosity then writeln('phase ',phase:1,temp:8:2);
        writev(w[1],'Homophonie',code:2);
        if ref[6,code] = 0 then create_clas(w[1],code,code,0,6) ;
        if phase = 2 then writeln(res,code : 3);
end ;
```

```
procedure lire_ligne(var a:phrase);
{  lit une ligne dans le fichier pf en en retirant les accents
   en outre convertit toute majuscule en minuscule  }

var     i       : integer ; {  indice du mot courant  }

begin
        readln(pf,original);
        desaccentuer(original,a);
        dcm(a) ;

{  si un mot se termine par une apostrophe
   alors supprimer l'apostrophe de ce mot  }

        for i:=1 to nbw
        do begin
              if w[i][length(w[i])] = ''''
              then w[i] := substr(w[i],1,length(w[i]) - 1 );
              j := index(w[i],'kwo') ;
              if j = 0 then j := index(w[i],'kwu');
              if j<>0
              then w[i] := 'k' + substr(w[i],3,length(w[i])-2);
           end ;

{  Si un mot se terminant par une voyelle
   précède un mot commencant par une voyelle
   alors supprimer la voyelle terminale du 1er mot  }

        for i:=2 to nbw
        do begin
              if ((w[i-1][length(w[i-1])] in voyelle )
              and (w[i][1] in voyelle))
              then begin
                      if length(w[i-1]) >= 2

                        then w[i-1] := substr ( w[i-1],1,length(w[i-1
                    end;
           end;
        c := '' ; for i:= 1 to nbw do c := c + w[i] ;
        d := '' ;
        for i:=1 to length(c)
        do if c[i] in voyelle then d := d + c[i] ;
end;
```

```
{  MAIN PROGRAM  }

begin
        open (pf,file_name:='paremy.run',history:=readonly);
        reset(pf);
        open (res,file_name := 'cl6.res',history:= new ) ;
        rewrite(res);

        read_clas ;
        i := 0;
        repeat i := i + 1
        until ((class[i].zone=6) or (i = nb_clas));
        if class[i].zone = 6
        then repeat kill_clas(i) until class[i].zone<> 6 ;

        write(' verbeux (O/N) ? ');
        if choice('onON') in ['o','O']
        then verbosity := true
        else verbosity := false ;
        write('nbs,nbv ? (22 30 conseillé)');
        readln(nbs,nbv);
        nbl := 0 ;

        while not eof(pf)
        do begin
                nbl := nbl + 1 ;
                lire_ligne(a);
                {  lit la ligne courante et tirer les accents  }
                {  et crée C la phrase telle qu'on la prononce
                   ainsi que D la suite des voyelles de C  }
                traitement_6(c);
                ecrire_6(c) ;
                build_code(1) ;
                alliteration(d);
                ecrire_6(d) ;
                build_code(2);
                if verbosity then writeln (code : 3);
           end;

        close(pf);
        close(res,save) ;
        save_clas;
end.
```

```
[ inherit('bt.env','string.env')]
program class7(input,output,res);

const   max_code7 = 80 ;
        undef = max_code7 ;

var     original: phrase ;
        { ligne courante de pf avec tous les accents  }
        a       : phrase ;
        { ligne courante de pf sans accent }
        ref7    : array [1..max_code7] of boolean ;
        { ref7[i] est vrai ssi la paremie courante
          contient une syllabe tendant
          a faire croire qu'elle appartient
          a la classe i de la zone 7  }
        nbref   : integer;
        { nombre de classes auquelles pourrait appartenir
                              la paremie courante  }
        total   : array [1..max_code7] of integer ;
        i,deb7  : integer ;
        nbl     : integer ;
        verbose : 1..4 ;
        res     : text  ;
        code    : integer ;
```

```
procedure traitement_7;

{  effectue le classement de la phrase a en rundi
   dans une ou plusieurs des classes de la zone [7]  }

var     i,j     : integer ;

begin
        for i:=1 to max_code7 do ref7[i] := false;

        if index(a,' ni ')<> 0 then ref7[11] := true ;
        if index(a,' n''')<>0 then ref7[11] := true ;
        if index(a,' si ')<> 0 then ref7[11] := true ;
        if index(a,' s''')<>0 then ref7[11] := true ;

        if index(a,' nta ')<>0 then ref7[12] := true ;
        if index(a,' nt''')<>0 then ref7[12] := true ;

        if index(a,'ri ')<> 0    then ref7[21] := true ;
        if index(a,' ar''')<> 0  then ref7[21] := true ;
        if index(a,'tari ')<>0   then ref7[21] := true ;
        if index(a,' hari ')<> 0 then ref7[22] := true ;

  {  EST-CE LA CLASSE 3 DE LA ZONE 6 ?  }

        if (index(original,' ~U')<> 0 ) then ref7[31] := true ;
        if (index(original,' Uw~')<> 0)  then ref7[31] := true ;
        if (index(original,' ~Ut')<> 0)  then ref7[31] := true ;
        if (index(original,' ~Ud')<> 0 )then ref7[31] := true ;

        {  Nd en debut de phrase suivi d'au moins quatre lettres  }

       if ((index(original,' Nd')<> 0 ) and (index(substr(a,2,4),' '
        then ref7[32] := true ;
        if (index(a,' sin') = 1 )then ref7[32] := true ;

        if ((index(original,' ~Ur')<> 0 )
        and (index(a,' uri') = 1))then ref7[33] := true ;
        if ((index(original,' ~Ir')<> 0 )
        and (index(a,' uri') = 1))then ref7[33] := true ;
        if (index(a,' ahari ') = 1 )then ref7[33] := true ;
        if (index(a,' akari ') = 1 )then ref7[33] := true ;
        if (index(a,' ivyari ')= 1 )then ref7[33] := true ;

 {  CLASSE 4 ?  }


       if ((index(a,' ni')< length(a)-4 ) and (index(a,' ni') <> 0 )
        then if (index(substr(a,index(a,' ni')+1,4),' ') = 0 )
             then ref7[41] := true ;

       if ((index(a,' si')< length(a)-4 ) and (index(a,' si') <> 0 )
        then if (index(substr(a,index(a,' si')+1,4),' ') = 0 )
             then ref7[41] := true ;

        if index(a,' ari ')<> 0 then ref7[21] := true ;
        if index(a,' ar''')<> 0 then ref7[21] := true ;
        if index(a,' ayo ')<>0 then ref7[42] := true ;
        if index(a,' ay''')<>0 then ref7[42] := true ;
```

```
        if index(a,' ico ')<>0 then ref7[42] := true ;
        if index(a,' ic''')<>0 then ref7[42] := true ;
        if index(a,' ivyo ')<>0 then ref7[42] := true ;
        if index(a,' ivy''')<>0 then ref7[42] := true ;
        if index(a,' ubwo ')<>0 then ref7[42] := true ;
        if index(a,' ubw''')<>0 then ref7[42] := true ;
        if index(a,' uko ')<>0 then ref7[42] := true ;
        if index(a,' uk''')<>0 then ref7[42] := true ;
        if index(a,' uwo ')<>0 then ref7[42] := true ;
        if index(a,' uw''')<>0 then ref7[42] := true ;


      if (ref7[42] and (index(a,' niko ')<>0)) then ref7[43] := tru

      if (ref7[42] and (index(a,' niwe ')<>0)) then ref7[43] := tru

        if index(a,' aho ')<>0 then ref7[44] := true ;
        if index(a,' iyo ')<>0 then ref7[44] := true ;
        if index(a,' iy''')<>0 then ref7[44] := true ;

{  CLASSE 5 ?  }

        if ((index(a,' ha ku')<>0)
        or (index(a,' ha gu')<>0))
        then ref7[51] := true ;
        if index(a,' hako ')<> 0 then ref7[51] := true ;

{  CLASSE 6 ?  }

        if ( index(a,' ngo ') <> 0)  then ref7[61] := true ;
        if ( index(a,' ngw ') <> 0)  then ref7[62] := true ;
        if (index(a,' ni wa') <> 0)  then ref7[63] := true ;

{  CLASSE 7 ?  }

        i := index(a,'ba ');
        if i=0 then i := index(a,'b''');
        if i<>0 then begin
                     j := i ;
                     repeat i:=i-1 until a[i]=' ';
                     if j-i-1 <= 3 then ref7[70]:= true;
                     end;

        code := undef ;
        ref7[undef] := true ;
        for i := 1 to max_code7
        do if ref7[i]
           then begin
                      nbref := nbref + 1 ;
                      ref7[undef]:= false ;
                      write(res,i:3);
                      total[i] := total[i] + 1 ;
                      code := i ;
                end;
        writeln(res) ;
end;
```

```
procedure ecrire_7;

var     i,j     : integer;

begin
        writeln('=============================================');
        writeln(original);
        writeln('N°',nbl:4,')    ',a);
        for i := 1 to max_code7
        do if ref7[i]
           then begin
                write(' code : ');
                write(i:2);
                j := 0 ;
                repeat j := j + 1 until class[j].zone = 7 ;
                writeln ('  (',class[(i div 10)+j-1].name,')');
                end;
end;


procedure conclusion_7;

var     i,j             : integer ;
        total_ref       : integer ;

begin
        write(chr(12));
        total_ref := 0 ;
        for i:=1 to max_code7 do total_ref := total_ref + total[i];
        for i := 1 to max_code7
        do if total[i]<>0 then writeln('ref7[',i,']:',total[i]:4);
end;
```

```
{  main PROGRAM  }
begin
        read_clas;
        deb7 := 0  ;
        repeat deb7 := deb7 + 1 until class[deb7].zone = 7 ;
        writeln('montrer les lignes ?1) Toutes ');
        writeln('                      2) Pas une');
        writeln('                      3) Sans référence');
        writeln('                      4) Avec référence');
        readln(verbose) ;
        open (pf,file_name:='paremy.run',history:=readonly);
        reset(pf);
        open (res,file_name:='cl7.res',history:=new);
        rewrite(res);

        for i:=1 to max_code7 do total[i] := 0;
        nbl := 0 ;
        while not eof(pf)
        do begin
                nbl := nbl + 1 ;
                readln(pf,original);
                desaccentuer(original,a);
                traitement_7;
                case verbose of
                1 : ecrire_7;
                3 : if code  = undef then ecrire_7 ;
                4 : if code <> undef then ecrire_7 ;
                end;
           end;
        conclusion_7 ;
        close(pf);
end.
```

```
[inherit('bt.env','string.env')]
program cl8(input,output,sf,kf,ff);

const   max_fix = 100 ;
        max_noeud = 20 ;
        max_kwd = 150 ;

type    fix     = varying[5] of char ;

        typ_noeud= array[1..max_fix,1..max_noeud] of fix ;

        typnat  = (inv,sub,verbe) ;
                                { nature du mot cle }

{
 si le mot-cle commence par $ c'est un inv(ariable) a ne pas decline
   sinon par defaut il s'agit d'un substantif declinable }

var     sf      : text;
                { contient les pre/suffixes en rundi }
        ff      : text;
                { contient la traduction des proverbes }
        res     : text; { contiendra les codes de la zone 8 }
        kf      : text; { contient le theme des mots cles }
        a       : phrase;
                { contiendra la phrase en rundi a analyser }
    traduction : phrase ;
                { contiendra sa traduction en francais }
        r       : real  ; { identifiant d'une paremie }
        p       : fix ;   { prefixe du mot analyse }
        t       : mot ;   { theme du mot analyse }
        sfix    : typ_noeud ;{contient les prefixes des substantifs}
    nb_sprefix : integer ;
    nb_ssuffix : integer ;
        nbps    : integer ;
                { nombre de prefixes possibles du substantif }
        i,j,k   : integer; { compteurs }
        ori,tra : array[1..max_fix] of fix ;
                { grammaire transformationnelle }
        nbp     : integer ;
              { NomBre de Prefixes possibles pour une structure no
        nbl     : integer ;
              { Numero De Ligne : numero de la paremie courante }
        code    : array[1..max_kwd] of boolean ;
                { codes de la paremie courante }
        kwd     : array[1..max_kwd] of mot ;
                { contiendra le theme des mots-cles }
        trad    : array[1..max_kwd] of mot ;
                { contiendra la traduction des mots-cles }
        renvoi  : array[1..max_kwd] of integer ;
                { renvoi[i] = code de kwd[i] }
        nat     : array[1..max_kwd] of typnat ;
                { nature du mot-cle }
        nbu     : integer ; { nombre de paremies sans mot cle }
        undef   : integer ;
                { code pour une paremie non identifiee }
        nb_kwd  : integer ; { nombre de mots-cles retenus }
        ok      : boolean ;
                { vrai ssi aucun mot-clé dans la parémie courante }
```

```
procedure read_kwd;

{
 lit les prefixes des mots_cles et met à jour le fichier des classes .

var     i,j     : integer ;
        line    : phrase ;

begin
        open(kf,file_name:='kwd.run',history:=readonly);
        reset(kf);
        i := 0 ;
        while not eof(kf)
        do begin
                readln(kf,line);
                i := i + 1 ;
                j := index(line,' ');
                if j<>0 then begin
                                kwd[i]:=substr(line,1,j-1);
                                if kwd[i][1] = '$'
                                then begin

                                    kwd[i] := substr(kwd[i],2,length
                                     nat[i] := inv ;
                                     end
                                else nat[i] := sub ;
                                trad[i]:='';
                                j:=length(line);
                                repeat trad[i]:=line[j] + trad[i] ;
                                                          j:=j-1
                                until line[j]=' '
                             end;
           end;
        close(kf);

        writeln;

       write('Nombre de mots-clés retenus (',i:2,' au maximum) : ');
        readln(nb_kwd);

        nb_kwd := nb_kwd + 1 ;

        kwd [nb_kwd] := 'smcl' ;
        trad[nb_kwd] := 'Sans mot-clé';
        undef := nb_kwd ;

        for i:=1 to nb_kwd do renvoi[i]:=i ;

{  par defaut le code d'un mot est le
   numero d'ordre de ce mot dans l'index  }

        for i:=1 to nb_kwd
        do  for j:=i+1 to nb_kwd {  j>i => renvoi[j]>=renvoi[i] }
                do begin
                        if equal(trad[i],trad[j])

                       then {  les traductions sont identiques => me
                        begin renvoi[j]:=renvoi[i] end ;
                   end ;
```

```
{  certaines classes seront vides mais éliminées par la suite ,
   lors de la construction de la table de contingence                }

        read_clas;

        for i:=1 to nb_clas
        do if class[i].zone = 8
           then repeat kill_clas(i)
           until ((class[i].zone<>8) or (i > nb_clas));
        print_clas ;

        for i:=1 to nb_kwd do create_clas(trad[i],i,i,0,8);
        save_clas ;
        print_clas ;

        if verbosity
        then begin

               writeln('N° mot-cle traduction          code%%

               writeln('=============================================
                for i:=1 to nb_kwd + 1

               do writeln(i:2,') ',kwd[i]:15,trad[i]:20,renvoi[i]:4,
             end;

end;
procedure lire_noeud(fn:mot;
          var noeud : typ_noeud;var nb_prefix,nb_suffix:integer) ;

var     i,j     :integer ;
        temp    : fix   ;

begin
        for i:=1 to max_fix
        do for j:=1 to max_noeud do noeud[i,j] := '      ';

        fn := fn + '.str';
        open (sf,file_name:=fn,history:=readonly);
        reset(sf);
        readln(sf,nb_prefix,nb_suffix);

        j := 0 ;
        i := 0 ;

        while not eof(sf)
        do begin
           readln(sf,temp);
           if index(temp,'$') <> 0
           then begin
                j := j + 1 ;
                i := 0  ;
                end
           else begin
                i := i + 1 ;
                noeud[i,j] := temp ;
                end;
           end;
        close(sf);
```

```
        nbps := i ;
end ;
function anal_sub_simple(var m : mot) :boolean ;

var     i,j     : integer ;
        rate    : boolean ;

                { vrai si le mot m n'appartient pas a la structure
        presence: integer ;

                { indice du dernier prefixe possible dans le noeud c
begin
        rate := (length(m)<=3) ;
        p := '';

        if not rate then
           begin
                i := 0 ;

             repeat i := i + 1 until ((i > nbps) or (index(m,sfix[
              if i <= nbps
              then begin
                   t := substr(m,length(sfix[i,1])+1,length(m)

                                                    -length(sfix[
                   p := p + sfix[i,1] ;
                   end
              else rate := true;
           end;

        if length(t)<2 then rate := true ;

        anal_sub_simple:= not rate ;
end;
function anal_sub_complex(var a:mot):boolean ;

var     i,j     : integer ;
        rep     : boolean ;
begin
        rep := false ;
        for i:=1 to nbp
        do if index(a,ori[i])=1
           then begin
                j := length(ori[i]) ;
                a := substr(a,j+1,length(a)-j);
                a := tra[i] + a ;
                if anal_sub_simple(a)

             then if rep then writeln(chr(7),'erreur dans anal_sub
                         else rep := true ;
              end ;
        anal_sub_complex := rep ;
end;
```

```
{  DETECTION DES MOTS-CLES }
begin
        write('B(eaucoup ou P(as de commentaires : ');
        if choice('BbPp') in ['B','b'] then verbosity := true
                                       else verbosity := false ;

        writeln;

        lire_noeud('substa',sfix,nb_sprefix,nb_ssuffix) ;

        read_kwd ;
        for i:=1 to nb_clas
        do  if class[i].zone = 8
            then class[i].effectif:=0;

        open (pf,file_name:='paremy.run',history:=readonly);
                                                        reset(pf);
        open (ff,file_name:='paremy.fra',history:=readonly);
                                                        reset(ff);
        open (sf,file_name:='substa.tfr',history:=readonly);
                                                        reset(sf);
        open (res,file_name:='cl8.res',history:=new);
                                                    rewrite(res);

        nbp := 0 ;

        while not eof(sf)
        do begin
                readln(sf,a) ;
                j := index(a,'=>');
                if j <> 0 then begin
                                nbp := nbp + 1 ;
                                ori[nbp]:= substr(a,1,j-1);

                               tra[nbp]:= substr(a,j+2,length(a)-(j+
                               end ;
           end;
        close(sf);


       if verbosity then for i:=1 to nbp do writeln(ori[i],'=>',tra[

        nbu := 0 ;
        nbl := 0;
        while not eof(pf)
        do begin
             for i:=1 to nb_kwd do code[i] := false ;
             nbl := nbl + 1 ;
             readln(pf,a);
             desaccentuer(a,a);
             readln(ff,traduction);
             desaccentuer(traduction,traduction);
             if verbosity
             then begin
                        writeln('==============================');
                        writeln(a);
                        writeln(traduction);
                        writeln ;
                  end;
```

```
            dcm(a) ;

{  decomposer la phrase en mot en pour chaque mot...  }

            for k:=1 to nbw
            do if length(w[k])>=4
               then begin t := '';

        {  le mot ne correspond-t-il pas à un mot invariant ?  }

               for j:=1 to nb_kwd
               do if ((nat[j] = inv) and ( equal(w[k],kwd[j])))
                  then t:=kwd[j];

        {  sinon est-il un nominal sans contraction ?  }

               if length(t) = 0 then
               if anal_sub_simple(w[k])
               then if verbosity then  writeln(p,'      ',t)

        {  sinon est-il un nominal avec contraction ?  }

               else if anal_sub_complex(w[k])
                       then if verbosity then  writeln(p,'      ',t,'
( }');



       {  si le mot courant repond a une de ces trois possibilites a

               if length(t)>0

              then begin              {  le theme du mot-cle est
                       for i:=1 to  nb_kwd
                       do if equal(t,kwd[i])
                          then begin
                               write(res,renvoi[i]:3);
                               code[i] := true ;
                               end;
                    end; {  of if  }


            {  aller voir dans la traduction du proverbe
               si se trouve la traduction d'un mot cle
               au singulier ou au pluriel  }

               for i:=1 to nb_kwd
               do if ((index(traduction,' ' + trad[i] + ' ' ) <> 0)

                 or  (index(traduction,' ' + trad[i] + 's' ) <> 0 )
                  then begin
                       code[i] := true ;
                       t:= kwd[i] ;
                       if verbosity then writeln(t,'    ',' {  }');
                       end ;
                  end ;


            ok := false ;
```

```
            for i:=1 to nb_kwd
            do if code[i] then begin
                                write(res,renvoi[i]:3);
                                ok := true ;
                                class[ref[8,renvoi[i]]].effectif :=

                               class[ref[8,renvoi[i]]].effectif + 1
                               end ;

            if not ok then writeln(res,undef:3)
                      else writeln(res);

            if not ok then nbu := nbu + 1 ;


            end; {  of while  }

        writeln('Sans mot-cle : ',nbu:4);
        close(res,save);
        writeln('suppression des sans mot clé');
        kill_clas(ref[8,undef]);
        save_clas ;
end.
```

```
[inherit('string.env','bt.env')]
program find_symbol(input,output);

const   max_symbol = 200 ;

var     a,b     : phrase ;
        r       : real ;
        i,j     : integer ;
        fin, deb: integer ;
        deb9    : integer ;
        { numero de la premiere classe de la zone 9 }
        let_fra : set of char ;
        fini    : boolean ;
        kf      : text ; { contient le theme des mots cles }
        res     : text ; { contiendra les codes }
        nbl     : integer ;
        { Numero De Ligne : numero de la paremie courante }
        code    : integer ;
                          { code minimum de la paremie courante }
        symbol  : array[1..max_symbol] of mot ;
        { contiendra le theme des mots-cles }
        trad    : array[1..max_symbol] of mot ;
        { contiendra la traduction des mots-cles }
        renvoi  : array[1..max_symbol] of integer ;
        { renvoi[i] = code de symbol[i] }
        nbu     : integer ; { nombre de paremies sans mot cle }
        undef   : integer ;
        { code pour une paremie non identifiee(pas de mot-cle }
        nb_symbol: integer ; { nombre de mots-cles retenus }


procedure read_symbol;

{ lit les mots symbolique et met à jour le fichier des classes }

var     i,j     : integer ;
        line    : phrase ;

begin
        open(kf,file_name:='symbol.run',history:=readonly);
        reset(kf);
        i:=0;

        while not eof(kf)
        do begin
                readln(kf,line);
                i := i + 1 ;
                j := index(line,' ');
                if j<>0 then begin
                                symbol[i]:=substr(line,1,j-1);
                                trad[i]:='';
                                j:=length(line);
                                repeat trad[i]:=line[j] + trad[i] ;
                                j:=j-1
                                until (line[j]=' ')
                             end;
           end;
        close(kf);
```

```
        nb_symbol := i ;
        nb_symbol := nb_symbol + 1 ;

        symbol [nb_symbol] := 'smcl' ;
        trad[nb_symbol] := 'Sans symbole';
        undef := nb_symbol ;

        for i:=1 to nb_symbol do renvoi[i]:=i ;

{  par defaut le code d'un mot est le
   numero d'ordre de ce mot dans l'index  }

        for i:=1 to nb_symbol
        do  for j:=i+1 to nb_symbol
        {  j>i => renvoi[j]>=renvoi[i] }
                do begin
                        if equal(trad[i],trad[j])
                        then
        {  les traductions sont identiques => meme code  }
                        begin renvoi[j]:=renvoi[i] end ;
                   end ;

{  certaines classes seront vides mais éliminées par la suite ,
   lors de la construction de la table de contingence              }

        read_clas;

        if class[nb_clas].zone = 9
        then repeat kill_clas(nb_clas)
        until class[nb_clas].zone<>9 ;

        for i:=1 to nb_symbol do create_clas(trad[i],i,i,0,9);

        if verbosity
        then begin
writeln('N°   Mot-cle              Traduction                Code');
writeln('=================================================');
for i:=1 to nb_symbol
do writeln(i:2,') ',symbol[i]:15,trad[i]:30,renvoi[i]:4);
    end;

end;


{  DETECTION DES MOTS SYMBOLIQUES }
begin
        read_symbol ;
        nb1 := 0 ;
        let_fra := min + maj + aac + eac + iac +
                   oac + uac + ['-'];

        open(res,file_name:='cl9.res',history:=new);
        rewrite(res) ;
        open(pf,file_name:='paremy.fra',history:=readonly);
        reset(pf) ;
        while not eof(pf)
        do begin
                nb1 := nb1 + 1 ;
                code := undef ;
```

```
                readln(pf,r,a);
                a:=substr(a,4,length(a)-3);
                if (index(a,'') <> 0)
                then
                begin
                dcm(a);
                for i:=1 to nbw
                do begin
                   if (length(w[i])>0 )
                   then begin
                        if (index(w[i],'') <> 0)
                        then begin
                             fin := index(w[i],'') - 1 ;

                            if w[i][fin] = 's' then fin := fin - 1 ;
                             deb := fin - 1 ;

                            fini := ((deb=0) or not (w[i][deb] in le
                             while not fini
                             do begin
                                deb := deb - 1 ;
                                if (deb=0)
                                then fini := true
                                else if not
                                         (w[i][deb] in let_fra)
                                         then fini := true ;
                                end ;
                             deb := deb + 1 ;
                             w[i]:=substr(w[i],deb,fin - deb + 1);
                             a := w[i] ;
                             capitalize(a) ;
                             for j :=1 to nb_symbol
                             do begin
                                if equal(a,symbol[j])
                                then if code = undef
                                        then code := renvoi[j]
                                        else begin
                                                write(res,code:3);
                                                code := renvoi[j]  ;
                                             end ;
                                end ;
                             end ; {  of if  }
                             end {  of if  }
                      end ; {  of for  }
                end; {  of if  }
                writeln(res,code:3) ;
           end;{  of while not eof  }

        write('Supprimer la classe sans mots-clés (O/N) : OUI !');
        nb_clas := nb_clas - 1 ;
        save_clas ;

        close(res,save) ;
        close(pf) ;

end.
```

```
[inherit('matrix.env','bt.env','string.env')]
program pg(input,output,res);

var     xy,z1,z2,l          : matrix ;
        h,largeur           : integer ;
        axe1,axe2           : integer ;
        zone1,zone2         : integer ;
        hscale,vscale,m     : double  ;
        fn                  : varying[80] of char ;
        screen              : array[-88..88,-70..70] of char ;
        lc                  : varying[30] of char ;
        ch                  : char ;
        i,j                 : integer ;
        res                 : text  ;
        sl                  : varying[max_size] of char;

function max(a : matrix) : double ;

{  retourne l'element maximum en valeur absolue de la matrice a  }

var     i,j     :integer ;
        m       :double  ;

begin
        m := abs(a.data[1,1]);
        for i:=1 to a.nbline do

       for j:=1 to a.nbcol do if abs(a.data[i,j])>m then m:=abs(a.da
        max := m ;
end;

procedure draw_axes ;

var     x,y     : integer ;

begin
        write('largeur et hauteur du graphique en caracteres : ');
        readln(lc);
        writeln ;
        if length(lc)>=3 then readv(lc,largeur,h);
        for x := -h to h do for y := -largeur to largeur
        do begin
                screen[x,y] := ' ' ;
                if x=0 then screen[x,y] := '-';
                if y=0 then screen[x,y] := '|';
           end;
end;
```

```
procedure build_graf(z:matrix ; axe1,axe2,mode : integer) ;

{  traduit en graphique les resultats de l'analyse factorielle
   axe1 et axe2 sont les numero des axes principaux
   mode = 1 => projection des profils lignes
   mode = 2 => projection des profils colonnes
   Z contient les projections des points          }

var     i,k,x,y : integer ;
        lc      : integer ;
        temp    : varying[2] of char ;

begin
        for i:=1 to z.nbline
        do begin
                x := round(vscale  z.data[i,axe2]);
                y := round(hscale  z.data[i,axe1]);
                if not (screen[x,y] in [' ','|','-','+'])

               then begin write(res,screen[x,y],' confondu avec
                            if mode = 2 then writeln(res,i)
                                        else writeln(res,sl[i]);
                     end;

                if not (screen[x,y] in [' ','|','-'])

               then begin write(screen[x,y],' confondu avec ') ;
                            if mode = 2 then writeln(i)
                                        else writeln(sl[i]);
                     end;

                if mode = 2 then begin
                                        writev(temp,i:1);
                                        for k:=1 to length(temp)
                                        do screen[x,y+k-1]:=temp[k];
                                 end
                            else screen[x,y] := sl[i];
           end;
end;

procedure cal_scale;

{
  calcule les echelles de projection des points de maniere a obtenir

  des points les plus distants possibles sans toutefois en perdre un

begin
        if max(col(z1,axe1)) > max(col(z2,axe1))
        then hscale := largeur/max(col(z1,axe1))
        else hscale := largeur/max(col(z2,axe1)) ;

        if max(col(z1,axe2)) > max(col(z2,axe2))
        then vscale := h/max(col(z1,axe2))
        else vscale := h/max(col(z2,axe2)) ;
 end;
```

```
procedure print_graf ;

var     i,j,k    : integer ;
        comment : array[-2..200] of varying[50] of char ;
        x,y      : integer ;

begin
        totalize(xy);
        for i:=-2 to 200 do comment[i]:='';
        j := 1;
        for i :=  1 to nb_clas
        do begin
              if class[i].zone = zone1
              then begin
                     writev(comment[j],s1[j]:2,' ',

                    class[i].name,'(',trunc(xy.data[j,xy.nbcol]):
                     j:=j+1 ;
                   end
           end;

        j:=j+2; k := j;
        for i:=1 to nb_clas
        do begin
              if class[i].zone = zone2
              then begin
                     writev(comment[j],j-k+1:2,' ',

               class[i].name,'(',trunc(xy.data[xy.nbline,j-k+1]):1,'
                     j:=j+1 ;
                   end ;
           end;

        comment[j] := '';
        writev(comment[j+1],'Inertie expliquée : ');
writev(comment[j+2],1.data[1,2]100
                              :5:2,'% + ',1.data[2,2]100 :5:2,'%');
writev(comment[j+3],1.data[2,3]100 :5:2,'% de l''inertie totale');

        i:=-2;
        for x:=-h to h
        do begin
              for y:=-largeur  to largeur do write(screen[x,y]);

             for y:=-largeur  to largeur do write(res,screen[x,y])
              writeln('  ',comment[i]);
              writeln(res,'  ',comment[i]);
              i := i + 1;
           end;
        for j:=-largeur to largeur+1 do write (' ');
        for j:=-largeur to largeur+1 do write (res,' ');
        writeln;
        writeln(res);
        write('echelle hor. = ',hscale/vscale :2:2,
              ' fois l''echelle verticale  ');
        write(res,'echelle hor. = ',hscale/vscale :2:2,
              ' fois l''echelle verticale  ');
        page(res);
end;
```

```
procedure kill_var;
{  supprime un ou plusieurs points-variables  }

var     i,j,k    : integer ;
        v        : array[1..10] of varying[5] of char;
        ch       : char ;
begin
        dcm(lc);
        for k:=1 to nbw
        do begin
                lc:=w[k];
                writeln('a tuer : ',lc);
                ch := lc[1];
                if ch in ['1'..'9']
                then begin
                        readv(lc,j);
                        kill_line(xy,j);
                        i := 0 ;
                        repeat i:=i+1 until class[i].zone = zone1 ;
                        j := i + j - 1 ;
                        kill_clas(j);
                      end;
                if ch in ['A'..'z']
                then begin
                        j := 0 ;

                       repeat j:= j+1 until ((sl[j] = ch) or (j>max_
                        kill_col(xy,j);
                        i := 0 ;
                        repeat i:=i+1 until class[i].zone = zone2 ;
                        j := i + j - 1 ;
                        kill_clas(j);
                     end;
        end;
end;


{  MAIN  }

begin
        axe1 := 2; axe2 := 3 ;
        write('Croisement entre quels criteres ? ');
        readln(zone1,zone2);
        writeln;
        writev(fn,'for0',zone1:1,zone2:1,'.dat');
        read_matrix(xy,fn) ;
        writev(fn,'pg',zone1:1,zone2:1,'.res');
        open(res,file_name:=fn,history:=new);rewrite(res);
        read_clas;
        lc := ' ';
        largeur:=45 ; h:=10;

      sl := 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';

repeat  begin
        if not (lc[1] in ['1','2','3','4','$'])
        then cpl(xy,z1,1);
```

```
        if not (lc[1] in ['1','2','3','4','$'])
        then cp2(xy,z1,1);

        draw_axes;
        cal_scale;
        build_graf (z1,axe1,axe2,1);
        build_graf (z2,axe1,axe2,2);
        print_graf ;


       write('variable à supprimer ? ("$" remontre le graphique):');
        readln(lc);

        if length(lc) > 0
        then case lc[1] of

       '1':for i:=1 to Z1.nbline do Z1.data[i,axe1] := -1.00  Z1.da

       '2':for i:=1 to Z1.nbline do Z1.data[i,axe2] := -1.00  Z1.da

       '3':for i:=1 to Z2.nbline do Z2.data[i,axe1] := -1.00  Z2.da

       '4':for i:=1 to Z2.nbline do Z2.data[i,axe2] := -1.00  Z2.da
        end;

        writeln ;

       if ((length(lc)<>0) and not (lc[1] in ['1','2','3','4','$']))
        then kill_var;
        end
until length(lc) = 0;

        writeln('Sauver les nouvelles donnees (O/N) :');
        if (choice('onON') in ['O','o'] )
        then begin
                writev(fn,'for0',zone1:1,zone2:1,'.dat');
                save_matrix(xy,fn);
                save_clas ;
             end;
end.
```

```
[inherit('string.env')]

program split(input,output,f1,f2,f3,f4,f5,f6,f7,f8,f9);

const   mf = 9 ;

var     a       : phrase ; {  ligne courante de pf  }

        b       : phrase ; {  ligne courante du fichier de sortie  }

        i       : 0..mf  ; {  compteur  }

        fn      : mot    ;

        fname   : array[1..mf] of name ;

        bi,bs   : array[1..mf] of integer ;

        f1,f2,f3,f4,f5,f6,f7,f8,f9 : text ;

        nb_files: integer ;


begin
        writeln('Ce programme éclate un fichier en plusieurs');
        writeln ;
        write('fichier à éclater (split.dat par defaut) : ');
        readln(fn);
        if length(fn) = 0 then fn:='split.dat';
        open(pf,file_name:=fn,history:=readonly);
        reset(pf);

        write('En combien de fichiers (entre 2 et 9) : ');
        nb_files := ord(choice('23456789')) - 48 ;

        for i:=1 to nb_files
        do begin
                write('fichier-résultat [',i:1,'] : ');
                readln(fname[i]);

               if length(fname[i]) = 0 then writev(fname[i],'c1',i:1
                case i of
                1 : open(f1,file_name:=fname[i],history:=new);
                2 : open(f2,file_name:=fname[i],history:=new);
                3 : open(f3,file_name:=fname[i],history:=new);
                4 : open(f4,file_name:=fname[i],history:=new);
                5 : open(f5,file_name:=fname[i],history:=new);
                6 : open(f6,file_name:=fname[i],history:=new);
                7 : open(f7,file_name:=fname[i],history:=new);
                8 : open(f8,file_name:=fname[i],history:=new);
                9 : open(f9,file_name:=fname[i],history:=new);
                end;
                case i of
                1 : rewrite (f1);
                2 : rewrite (f2);
                3 : rewrite (f3);
                4 : rewrite (f4);
                5 : rewrite (f5);
                6 : rewrite (f6);
```

```
                7 : rewrite (f7);
                8 : rewrite (f8);
                9 : rewrite (f9);
                end;


              writeln('Entrez maintenant les coordonnées de la part

              writeln('de ',fn,' que vous voulez écrire dans ',fnam
               write('position du premier _dernier caratère : ');
               readln(bi[i],bs[i]) ;
               writeln ;
          end;

        write('Qu''ajouter au début de chaque fichier : ');
        readln(fn) ;
        while not eof(pf)
        do begin
              readln(pf,a) ;
              for i:=1 to nb_files
              do begin
                    if bs[i] > 0

                   then b := substr(a,bi[i],bs[i] - bi[i] + 1 )

                   else b := substr(a,bi[i],length(a) - bi[i] +
                    b := fn + b ;
                    case i of
                    1 : writeln (f1,b);
                    2 : writeln (f2,b);
                    3 : writeln (f3,b);
                    4 : writeln (f4,b);
                    5 : writeln (f5,b);
                    6 : writeln (f6,b);
                    7 : writeln (f7,b);
                    8 : writeln (f8,b);
                    9 : writeln (f9,b);
                    end;
                 end;
          end ;

        for i:=1 to nb_files
        do case i of
              1 : close (f1,save);
              2 : close (f2,save);
              3 : close (f3,save);
              4 : close (f4,save);
              5 : close (f5,save);
              6 : close (f6,save);
              7 : close (f7,save);
              8 : close (f8,save);
              9 : close (f9,save);
              end;

        close(pf);
end.
```

N°	CRITERE	NOM DE LA CLASSE	EFFECTIF	CODES ... A ...	
1	1	Avoir	462	1	462
2	1	Activité	385	463	847
3	1	Savoir	420	848	1267
4	1	Etre	451	1268	1718
5	1	Cohésion	409	1719	2127
6	1	Solidarité	200	2128	2327
7	1	Insociabilité	490	2328	2817
8	1	Animosité	389	2818	3207
9	1	Autorité	65	3208	3272
10	1	Protection	215	3273	3487
11	1	Domination	148	3488	3635
12	1	Abus de pouvoir	77	3636	3712
13	1	Destin	124	3713	3836
14	1	Aléas du destin	296	3837	4132
15	1	Destin individuel	79	4133	4211
16	1	Fatalité	245	4212	4456
17	2	Proverbe indicatif	3169	10	10
18	2	Proverbe directif	115	11	11
19	2	Maxime	61	20	20
20	2	Aphorisme	1722	30	30
21	2	Locution proverbiale	57	40	41
22	2	Dicton	9	50	52
23	2	Adage	26	60	61
24	2	Apophtegme et autre	49	90	99
25	3	Substitution concrète	377	10	10
26	3	Substitution abstraite	2510	11	11
27	3	Substitution partielle	1071	20	20
28	3	Parémie en langage clair	497	30	30
29	4	Binaire simple	221	10	10
30	4	Binaire double	1450	11	11
31	4	Tercet	16	12	12
32	4	Barform	9	13	13
33	4	Autre procedé (si-niko)	2155	14	14
34	4	Monostiche	73	20	20
35	4	Autre procedé(21)	332	21	21
36	4	Asymétrie	199	30	30
37	5	Evidence	159	10	15
38	5	Vraisemblance	408	20	22
39	5	Assertorique	1386	30	32
40	5	Croyance	46	40	42
41	5	Affirmation gratuite	1	50	53

N°	CRITERE	NOM DE LA CLASSE	EFFECTIF	CODES ... A ...	
42	6	Homophonie 0	335	0	0
43	6	Homophonie 1	226	1	1
44	6	Homophonie 2	213	2	2
45	6	Homophonie 3	74	3	3
46	6	Homophonie 4	23	4	4
47	6	Homophonie10	73	10	10
48	6	Homophonie11	127	11	11
49	6	Homophonie12	178	12	12
50	6	Homophonie13	120	13	13
51	6	Homophonie14	49	14	14
52	6	Homophonie20	57	20	20
53	6	Homophonie21	78	21	21
54	6	Homophonie22	115	22	22
55	6	Homophonie23	103	23	23
56	6	Homophonie24	49	24	24
57	6	Homophonie30	14	30	30
58	6	Homophonie31	12	31	31
59	6	Homophonie32	30	32	32
60	6	Homophonie33	45	33	33
61	6	Homophonie34	39	34	34
62	6	Homophonie40	4	40	40
63	6	Homophonie41	3	41	41
64	6	Homophonie42	4	42	42
65	6	Homophonie43	14	43	43
66	6	Homophonie44	15	44	44
67	7	proposition nominale	861	10	19
68	7	proposition nomino-verbale	434	20	29
69	7	proposition verbale	649	30	39
70	7	proposition subordonnee	766	40	49
71	7	subordonnee sans conjonction	31	50	59
72	7	subordonnee conjonctionnelle	121	60	69
73	7	proposition avec verbe aux.	305	70	70
74	7	proposition indefinie	2041	80	80

N°	CRITERE	NOM DE LA CLASSE	EFFECTIF	CODES ... A ...	
75	8	homme	461	1	1
76	8	enfant	484	2	2
77	8	chance	31	4	4
78	8	individu	198	5	5
79	8	vache	207	6	6
80	8	chien	106	7	7
81	8	verite	106	8	8
82	8	roi	235	9	9
83	8	ventre	272	10	10
84	8	enclos	119	11	11
85	8	femme	158	12	12
86	8	pauvre	49	13	13
87	8	eau	121	14	14
88	8	mal	148	16	16
89	8	pate	82	17	17
90	8	biere	70	18	18
91	8	coeur	102	19	19
92	8	manger	75	20	20
93	8	ruse	19	21	21
94	8	chemin	70	22	22
95	8	lait	91	23	23
96	8	geniteur	56	24	24
97	8	maison	147	25	25
98	8	bouche	77	26	26
99	8	aieul	57	27	27
100	8	mere	255	28	28
101	8	pere	60	30	30
102	8	mort	71	31	31
103	8	kraal	64	32	32
104	8	force	54	33	33
105	8	pluie	43	34	34
106	8	viande	72	36	36
107	8	yeux	66	37	37
108	8	perdrix	53	38	38
109	8	nuit	89	39	39
110	8	malchanceux	18	41	41
111	8	neant	57	42	42

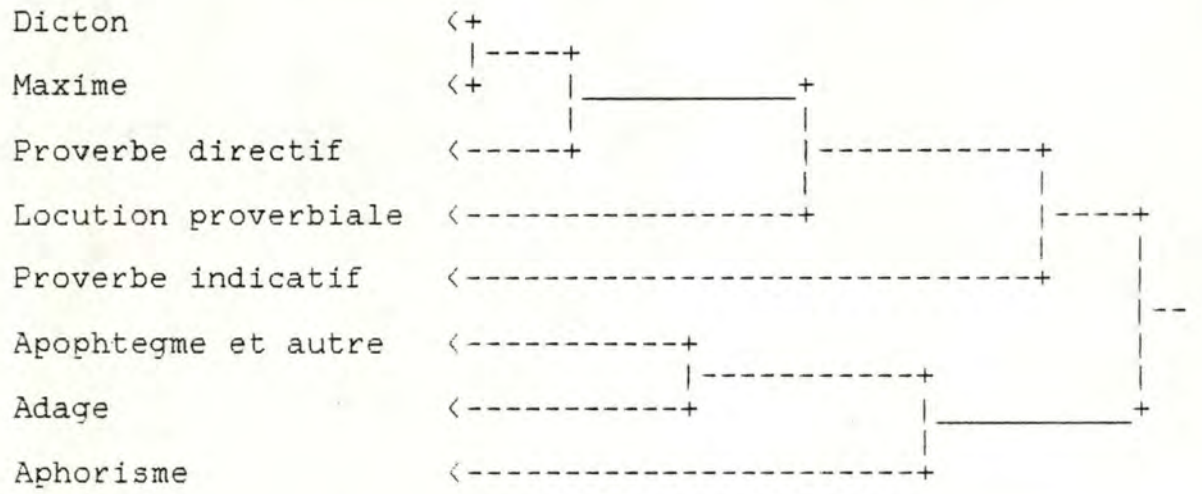
N°	CRITERE	NOM DE LA CLASSE	EFFECTIF	CODES ... A ...	
112	9	BANANE,BARATTE	8	1	1
113	9	MARMITE,BATON,SUIE	29	4	4
114	9	BELLE-MERE	9	5	5
115	9	BIERE,BEURRE	44	6	6
116	9	BOUCLIER	6	8	8
117	9	CHEVRE,GRENIER	36	9	9
118	9	CHIEN	93	10	10
119	9	COEUR	21	11	11
120	9	DENT	13	12	12
121	9	ELEPHANT	8	13	13
122	9	EPAR,OEIL	32	14	14
123	9	FAMINE	17	15	15
124	9	TAUREAU,FOUDRE	19	16	16
125	9	PROVERBE,MOU,FOURMI	29	17	17
126	9	GALE	19	18	18
127	9	GENDRE	2	19	19
128	9	VALET	13	21	21
129	9	PYGMEE,HYENE	24	22	22
130	9	JAMBE	15	23	23
131	9	JOUR	7	24	24
132	9	LAIT	61	25	25
133	9	VEAU,LANCE	30	26	26
134	9	LEOPARD	4	27	27
135	9	MAQUIS	8	28	28
136	9	MARTEAU	6	30	30
137	9	MEULE	2	31	31
138	9	NUIT	45	34	34
139	9	PLUIE	35	36	36
140	9	PORTE	3	37	37
141	9	SERPENT,PRECIPICE	7	38	38
142	9	RAT	8	41	41
143	9	ROI	118	42	42
144	9	ROSEE	6	43	43
145	9	SEL	8	44	44
146	9	SORGHO	9	46	46
147	9	SPATULE	3	47	47
148	9	TABAC	3	49	49
149	9	TAMBOUR	17	50	50
150	9	TESTICULE	8	52	52
151	9	TUTSI	7	53	53
152	9	VAN,VENTRE	19	54	54
153	9	VENT	7	56	56

Table de contingence en valeurs absolues

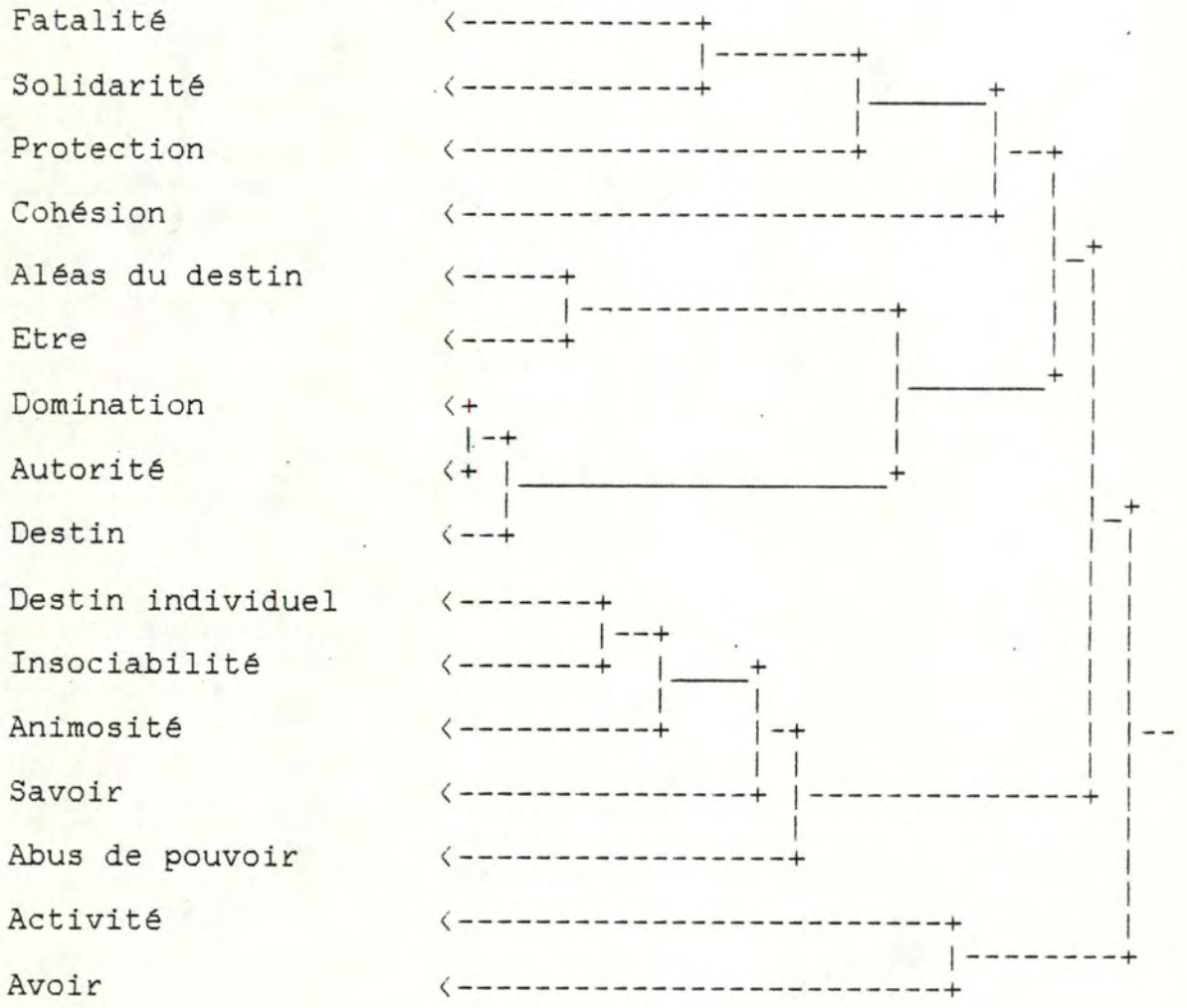
Avoir	271	25	15	127	18	5	1	0	462
Activité	240	15	8	98	20	0	4	0	385
Savoir	271	15	5	123	0	0	1	5	420
Etre	286	3	3	154	0	1	0	4	451
Cohésion	211	1	0	179	4	0	11	3	409
Solidarité	123	3	6	61	2	0	3	2	200
Insociabilité	324	11	1	148	0	0	1	5	490
Animosité	254	11	0	123	0	1	0	0	389
Autorité	37	0	0	28	0	0	0	0	65
Protection	123	3	5	74	0	0	0	10	215
Domination	81	0	1	66	0	0	0	0	148
Abus de pouvoir	58	0	0	17	0	2	0	0	77
Destin	76	0	0	48	0	0	0	0	124
Aléas du destin	193	0	0	99	0	0	0	4	296
Destin individuel	61	0	0	17	0	0	1	0	79
Fatalité	148	0	5	88	0	0	0	4	245
Total colonnes	2757	87	49	1450	44	9	22	37	4455

Proverbe indicatif
Proverbe directif
Maxime
Aphorisme
Locution proverbiale
Dicton
Adage
Apophtegme et autre
Total lignes

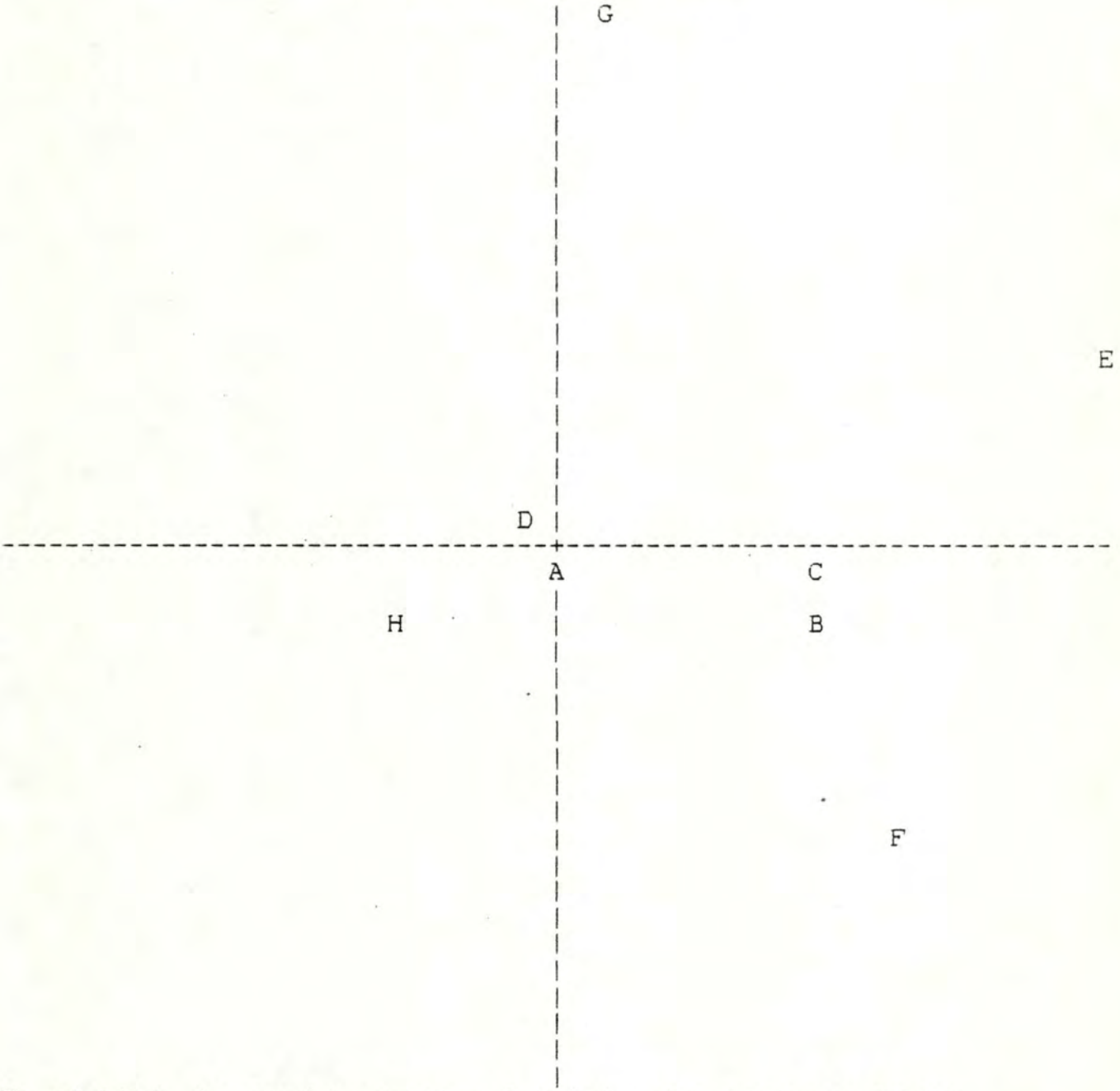
Hiérarchisation sur base de la variance



Hiérarchisation sur base de la variance



Analyse des correspondances

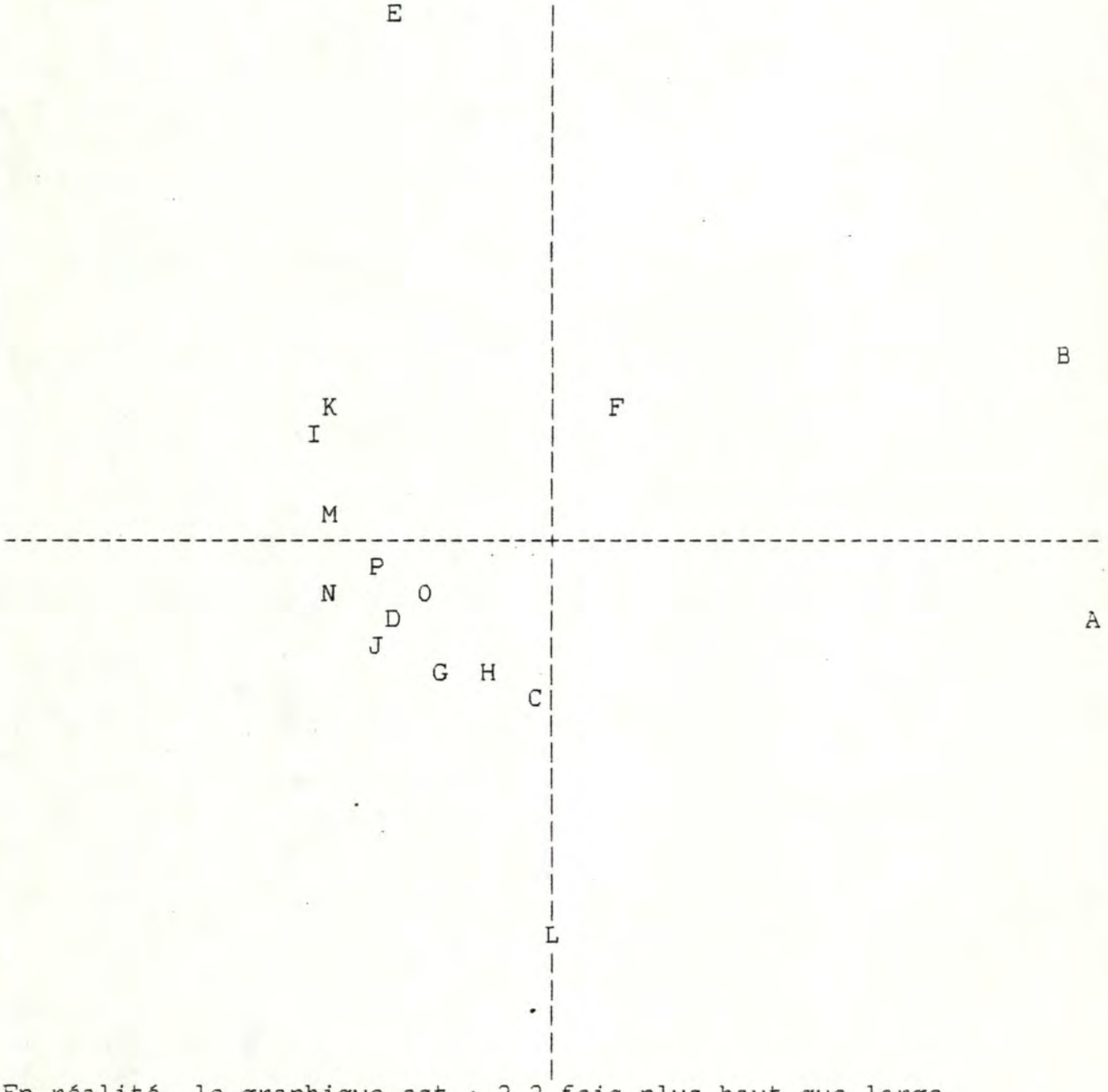


En réalité, le graphique est : 2.4 fois plus haut que large

A:Proverbe indicatif	2757	B:Proverbe directif	87
C:Maxime	49	D:Aphorisme	1450
E:Locution proverbiale	44	F:Dicton	9
G:Adage	22	H:Apophtegme et autre	37

Inertie expliquée : 59.84% + 23.95% = 83.79% de l'inertie totale

Analyse des correspondances



En réalité, le graphique est : 2.2 fois plus haut que large

A:Avoir	462	B:Activité	385	C:Savoir	420
D:Etre	451	E:Cohésion	409	F:Solidarité	200
G:Insociabilité	490	H:Animosité	389	I:Autorité	65
J:Protection	215	K:Domination	148	L:Abus de pouvoir	77
M:Destin	124	N:Aléas du destin	296	O:Destin individue	79
P:Fatalité	245				

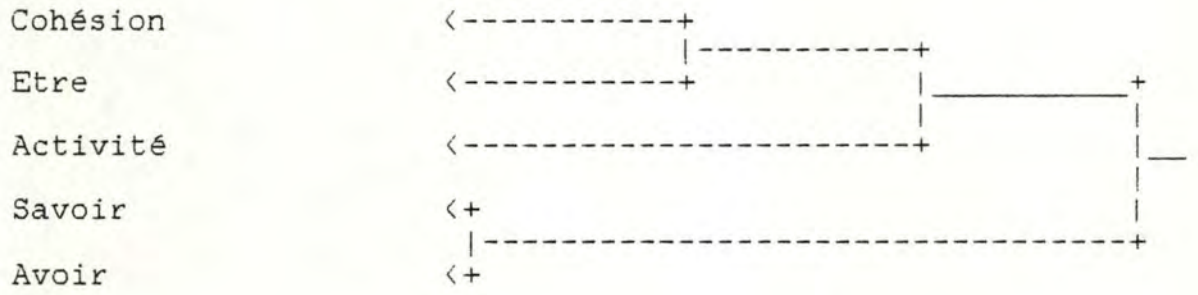
Inertie expliquée : 59.84% + 23.95% = 83.79% de l'inertie totale

Table de contingence en valeurs absolues

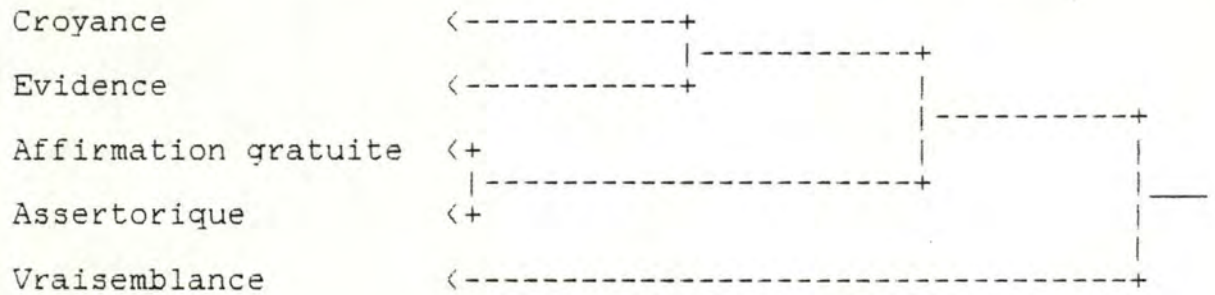
Avoir	22	146	290	4	462
Activité	59	75	243	8	385
Savoir	27	104	279	10	420
Etre	26	45	371	8	450
Cohésion	25	38	203	16	282
Solidarité	0	0	0	0	0
Insociabilité	0	0	0	0	0
Animosité	0	0	0	0	0
Autorité	0	0	0	0	0
Protection	0	0	0	0	0
Domination	0	0	0	0	0
Abus de pouvoir	0	0	0	0	0
Destin	0	0	0	0	0
Aléas du destin	0	0	0	0	0
Destin individuel	0	0	0	0	0
Fatalité	0	0	0	0	0
Total colonnes	159	408	1386	46	1999

Evidence
Vraisemblance
Assertorique
Croyance
Total lignes

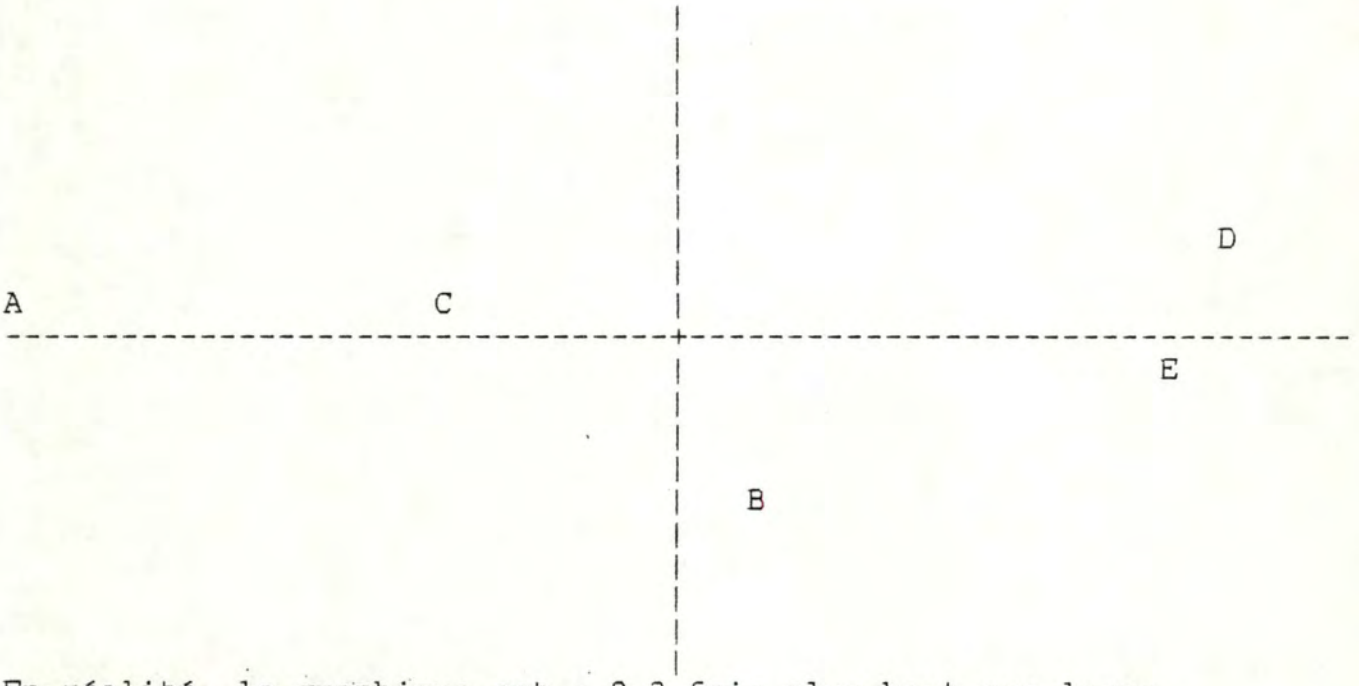
Hiérarchisation sur base de la variance



Hiérarchisation sur base de la variance



Analyse des correspondances

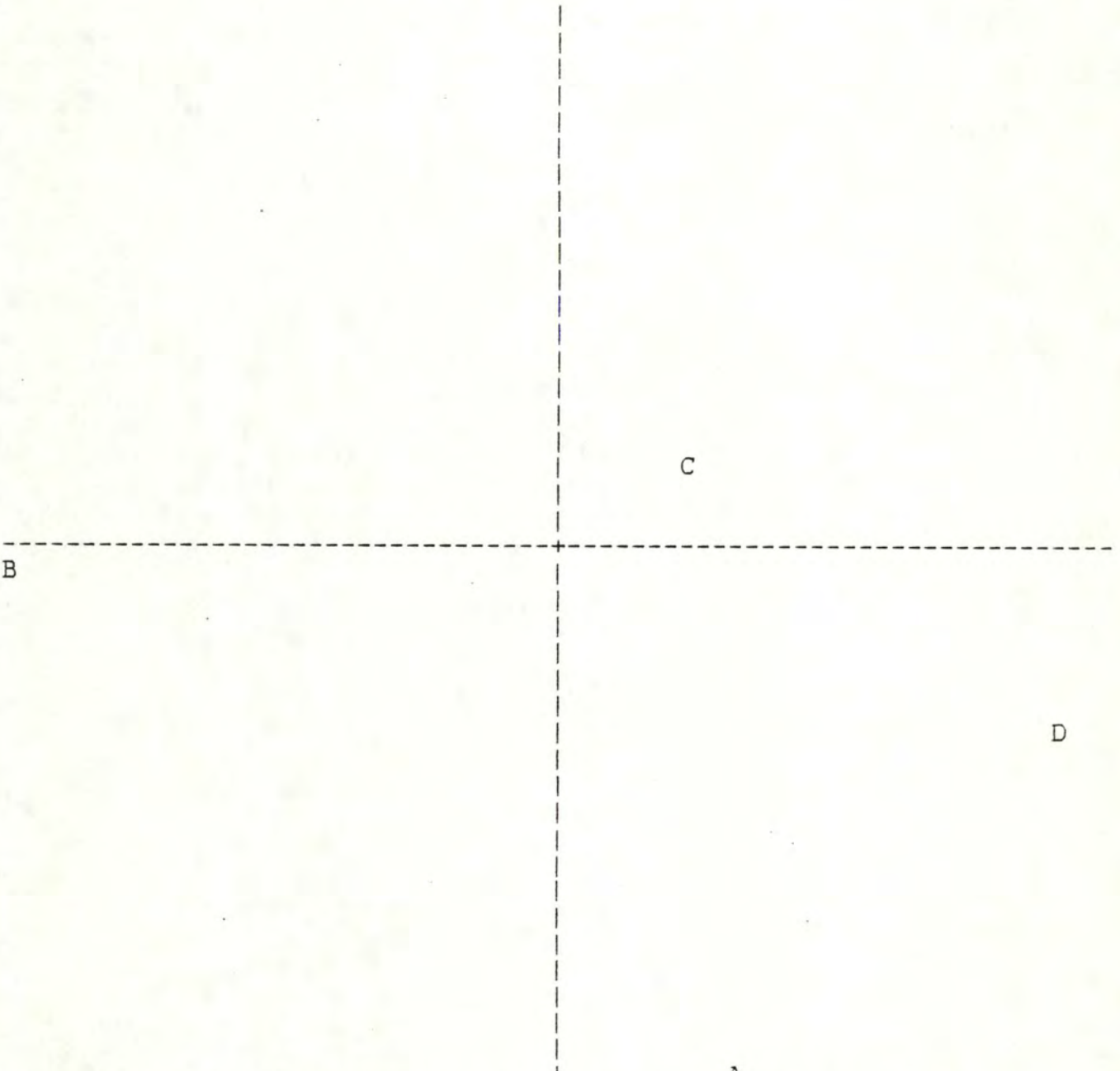


En réalité, le graphique est : 8.3 fois plus haut que large

A:Avoir	462	B:Activité	385	C:Savoir	420
D:Etre	450	E:Cohésion	282		

Inertie expliquée : 60.49% + 29.55% = 90.04% de l'inertie totale

Analyse des correspondances



En réalité, le graphique est : 2.8 fois plus haut que large

A:Evidence	159	B:Vraisemblance	408
C:Assertorique	1386	D:Croyance	46

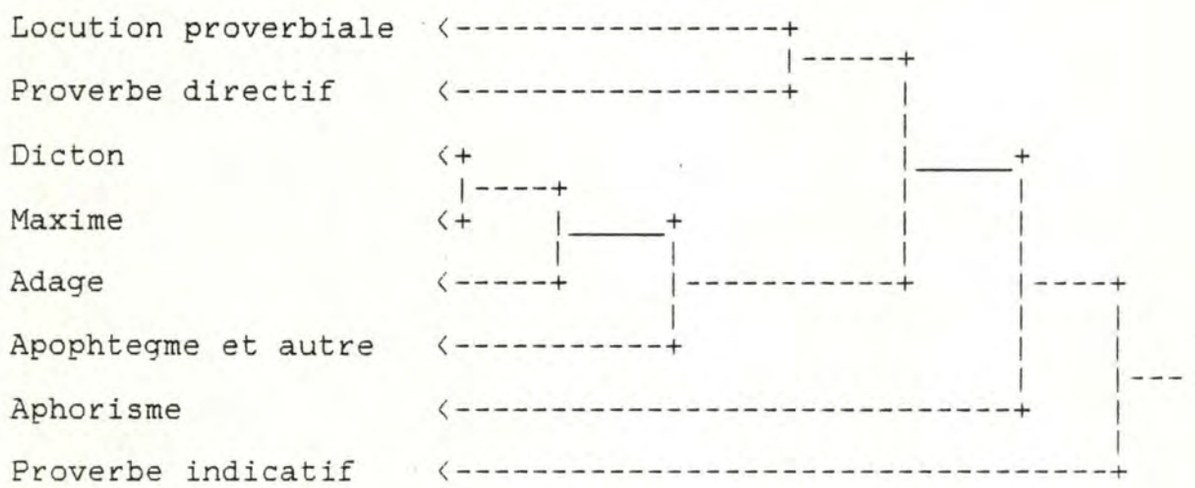
Inertie expliquée : 60.49% + 29.55% = 90.04% de l'inertie totale

Table de contingence en valeurs absolues

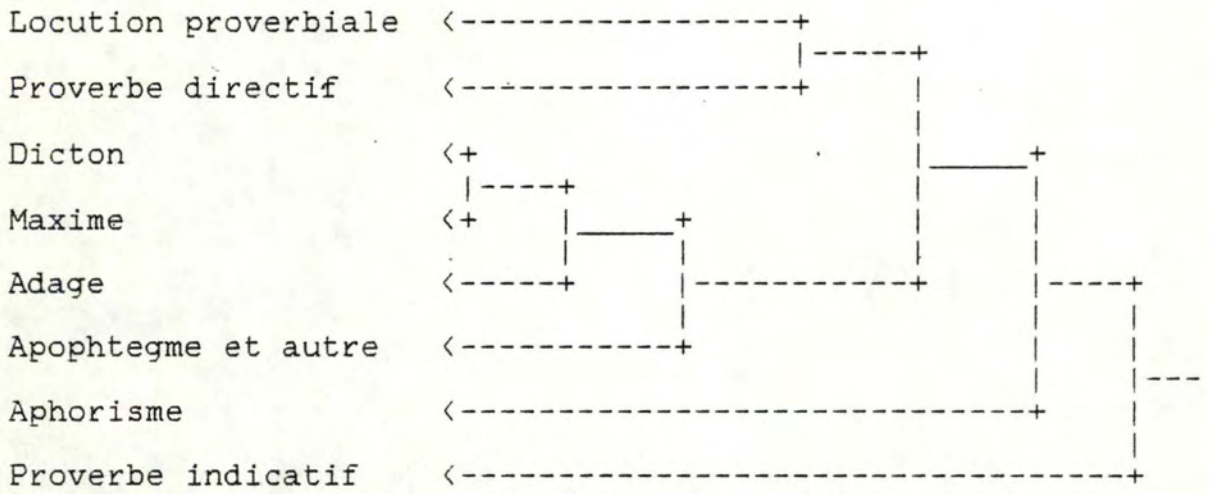
Proverbe indicatif	75	1003	7	6	1279	46	232	109	2757
Proverbe directif	5	33	1	0	28	1	15	4	87
Maxime	9	15	1	0	15	3	2	4	49
Aphorisme	119	364	7	1	792	21	74	72	1450
Locution proverbiale	1	15	0	2	16	0	7	3	44
Dicton	0	4	0	0	2	2	1	0	9
Adage	8	7	0	0	5	0	1	1	22
Apophtegme et autre	4	9	0	0	18	0	0	6	37
Total colonnes	221	1450	16	9	2155	73	332	199	4455

Binaire simple
Binaire double
Tercet
Barform
Autre procedé (si-niko)
Monostiche
Autre procedé(21)
Asymétrie
Total lignes

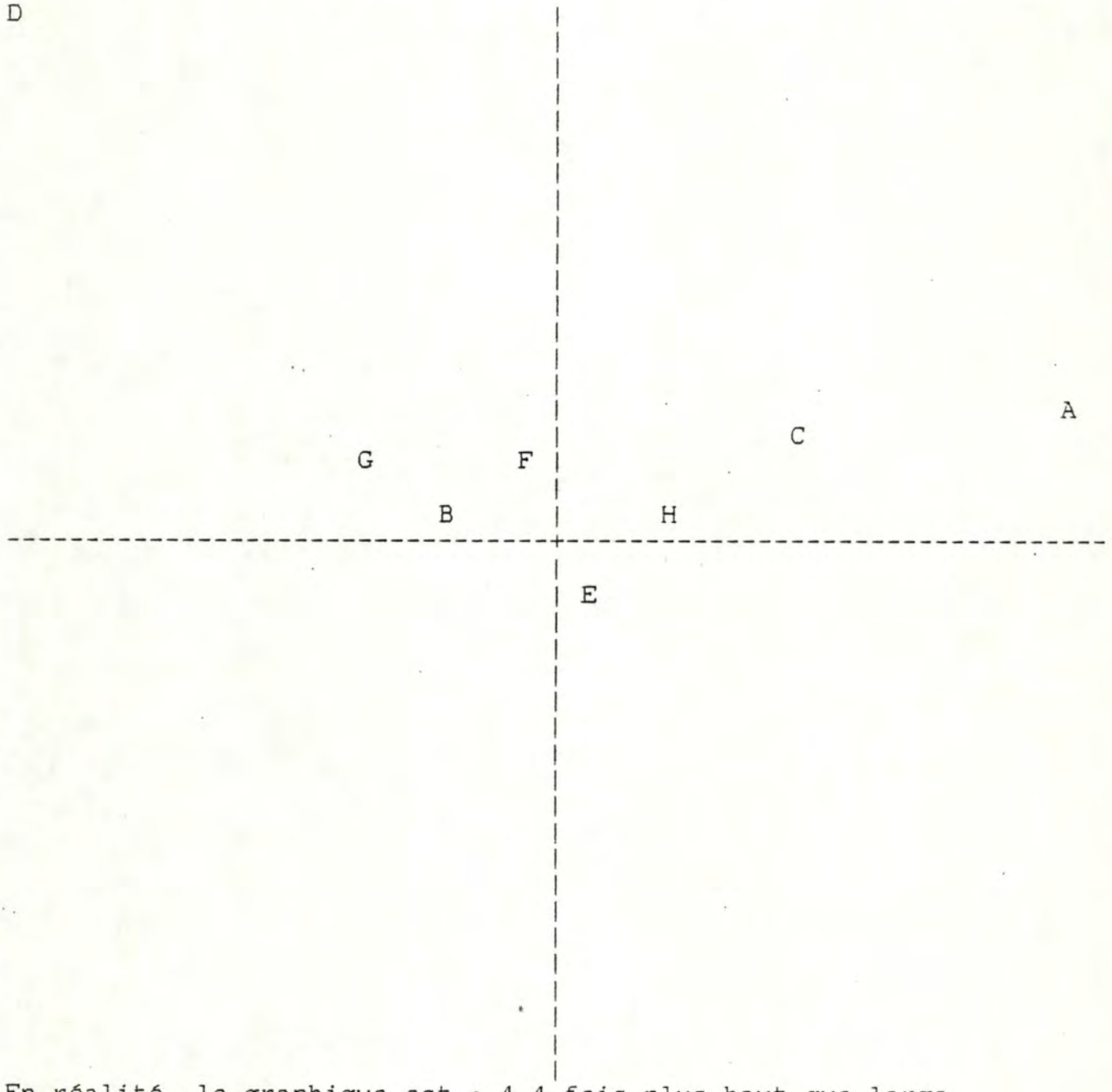
Hiérarchisation sur base de la variance



Hiérarchisation sur base de la variance



Analyse des correspondances

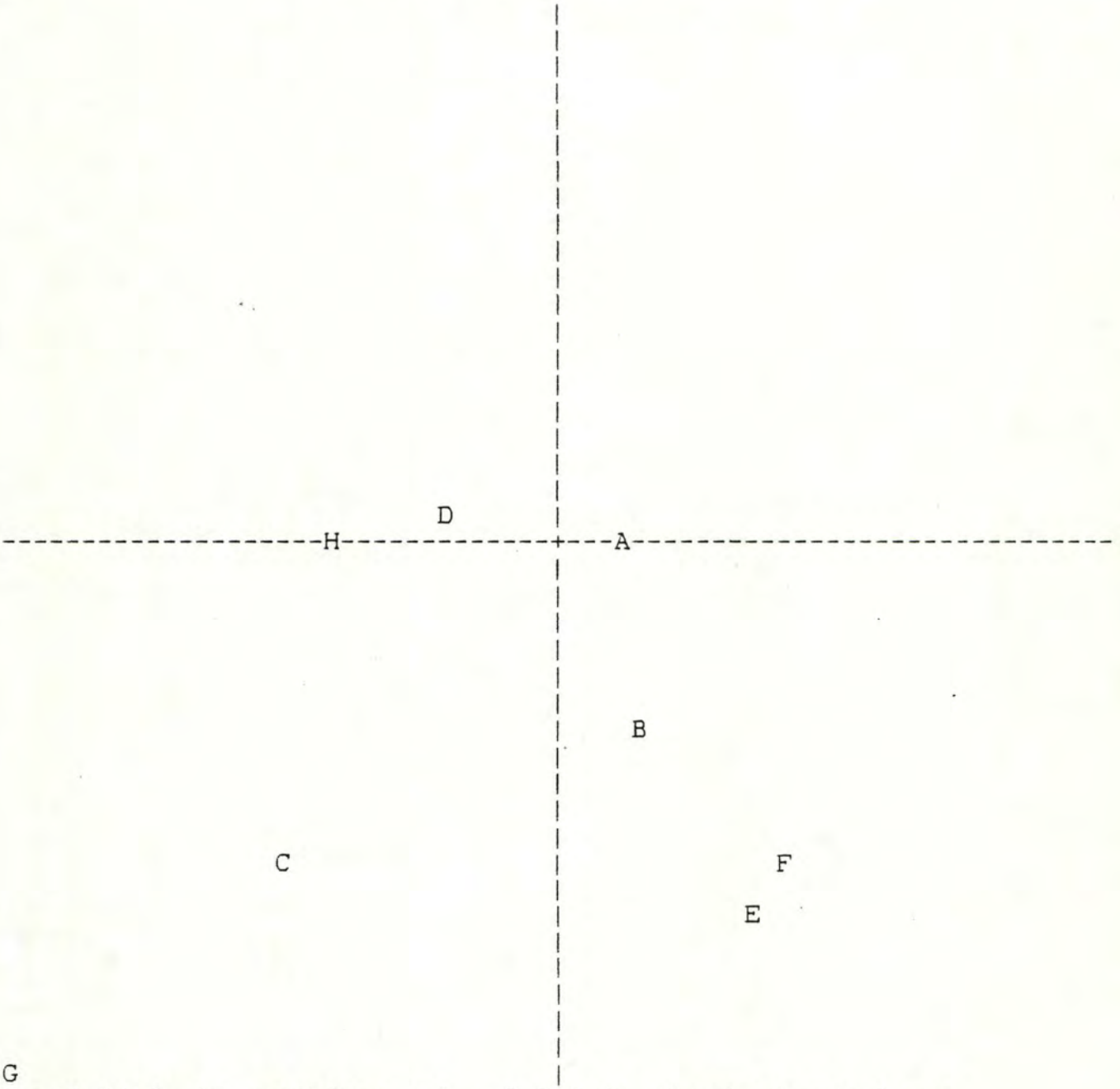


En réalité, le graphique est : 4.4 fois plus haut que large

A:Binaire simple	221	B:Binaire double	1450
C:Tercet	16	D:Barform	9
E:Autre procedé (si-niko)	2155	F:Monostiche	73
G:Autre procedé(21)	332	H:Asymétrie	199

Inertie expliquée : 64.37% + 19.14% = 83.51% de l'inertie totale

Analyse des correspondances



En réalité, le graphique est : 2.3 fois plus haut que large

A:Proverbe indicatif	2757	B:Proverbe directif	87
C:Maxime	49	D:Aphorisme	1450
E:Locution proverbiale	44	F:Dicton	9
G:Adage	22	H:Apophtegme et autre	37

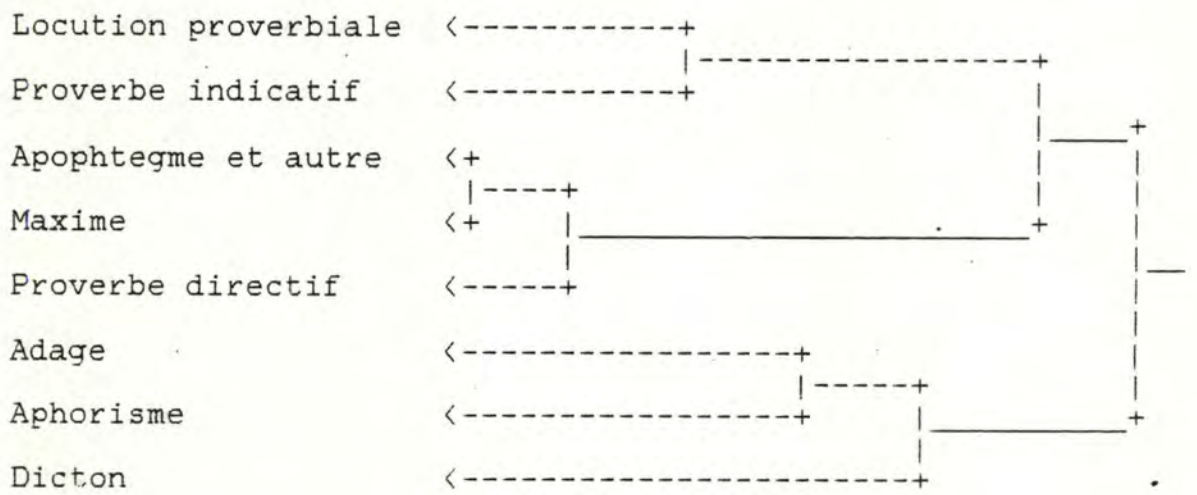
Inertie expliquée : 64.37% + 19.14% = 83.51% de l'inertie totale

Table de contingence en valeurs absolues

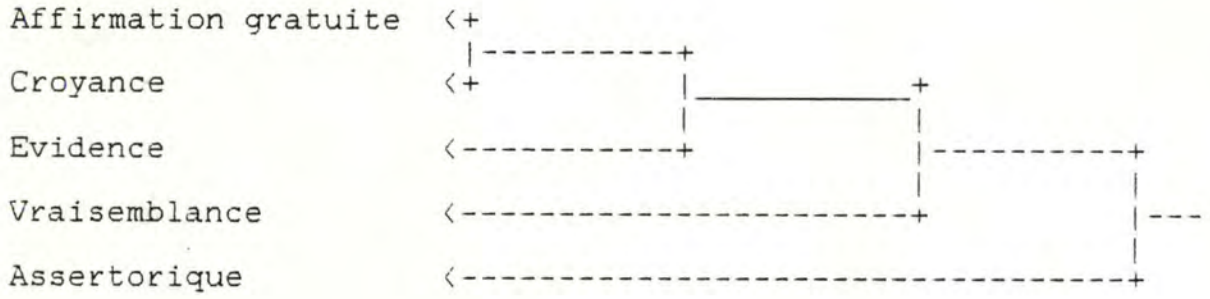
Proverbe indicatif	134	245	832	26	1237
Proverbe directif	0	1	58	0	59
Maxime	0	0	31	0	31
Aphorisme	21	155	408	12	596
Locution proverbiale	4	2	33	3	42
Dicton	0	0	5	0	5
Adage	0	5	7	5	17
Apophtegme et autre	0	0	12	0	12
Total colonnes	159	408	1386	46	1999

Evidence
Vraisemblance
Assertorique
Croyance
Total lignes

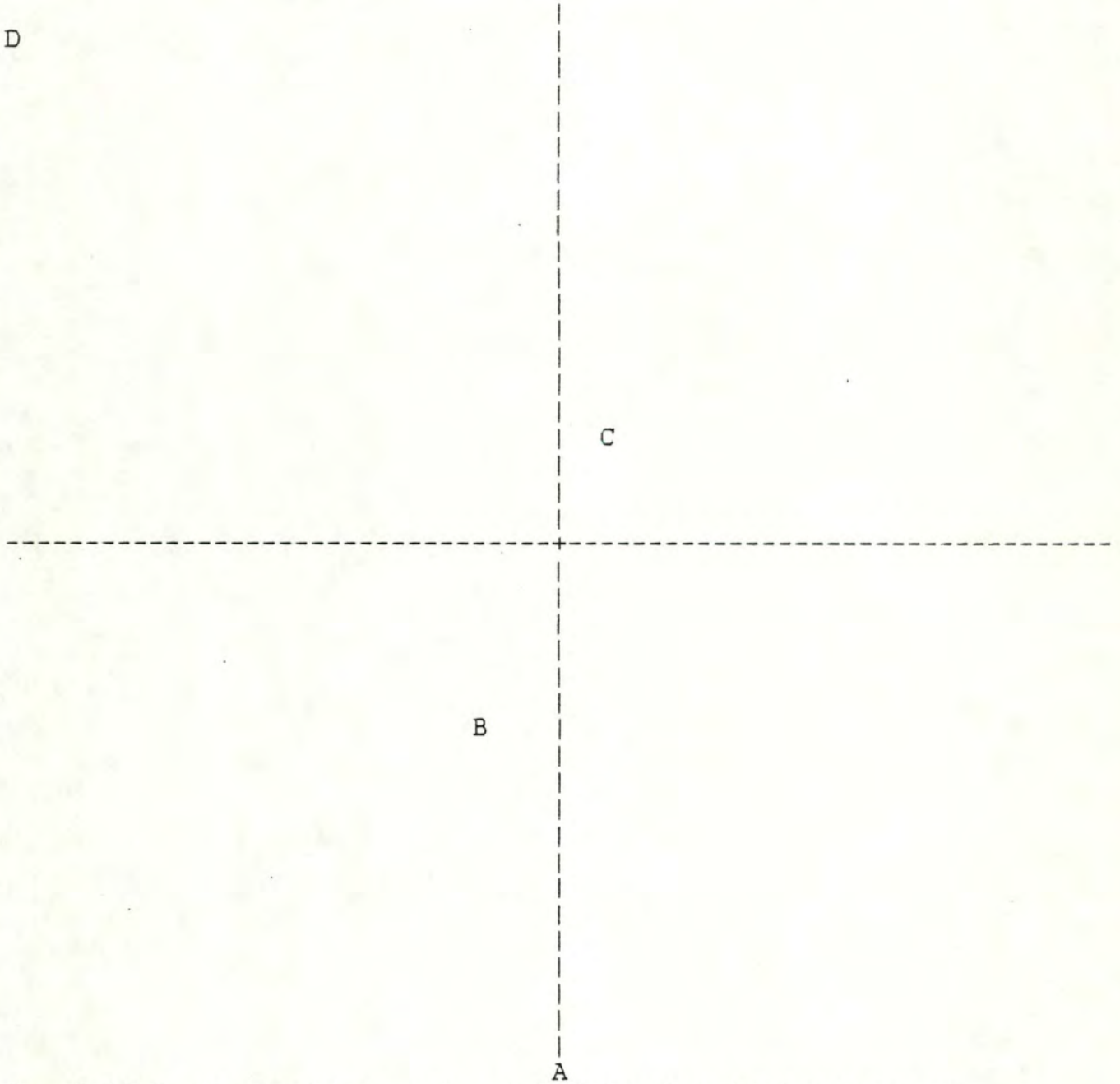
Hiérarchisation sur base de la variance



Hiérarchisation sur base de la variance



Analyse des correspondances



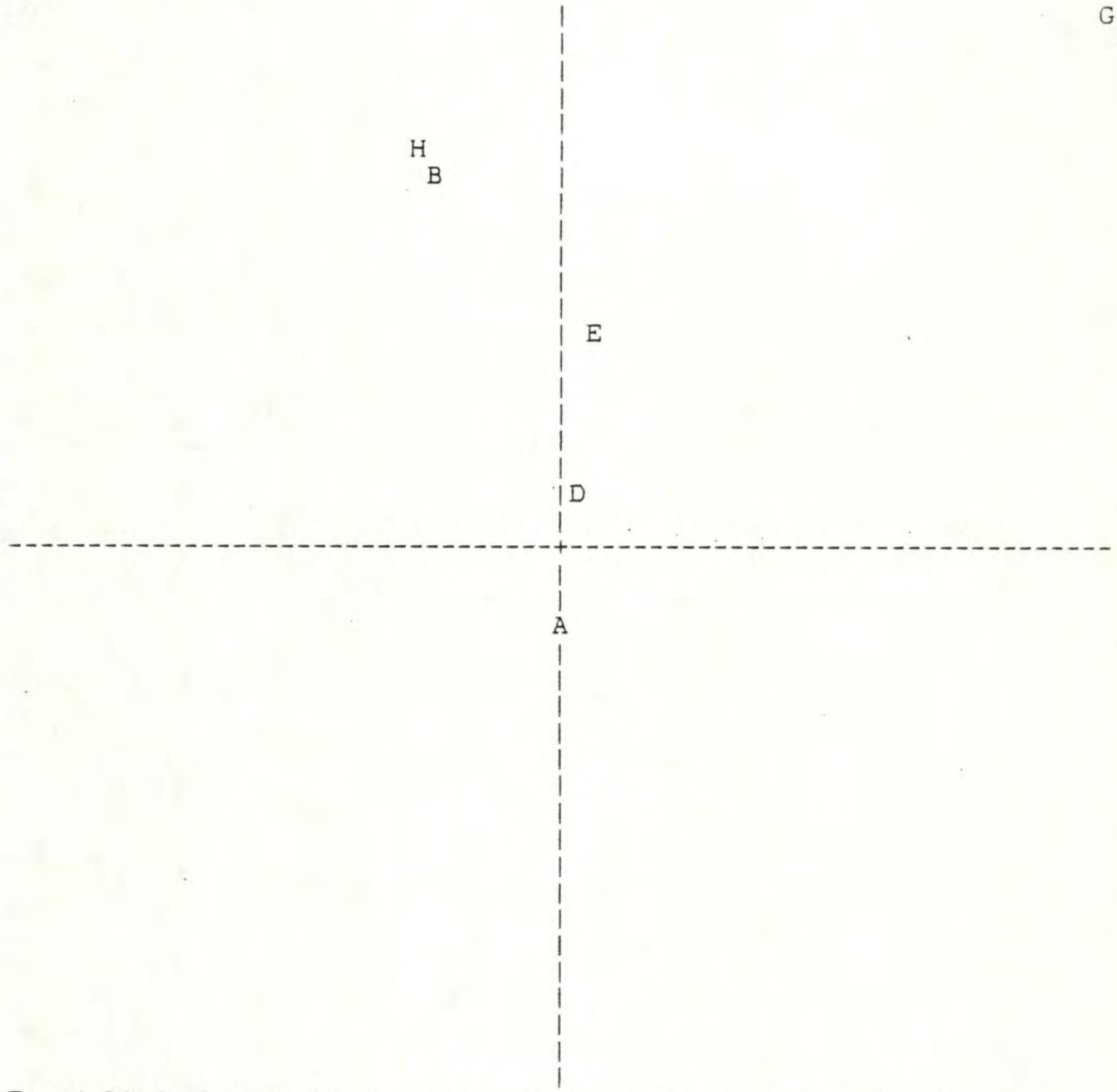
En réalité, le graphique est : 1.3 fois plus haut que large

A:Evidence	159	B:Vraisemblance	408
C:Assertorique	1386	D:Croyance	46

Inertie expliquée : 46.72% + 29.23% = 75.95% de l'inertie totale

Analyse des correspondances

C = F ; F = H ;



En réalité, le graphique est : 1.2 fois plus haut que large

A:Proverbe indicatif	1237	B:Proverbe directif	59
C:Maxime	31	D:Aphorisme	596
E:Locution proverbiale	42	F:Dicton	5
G:Adage	17	H:Apophtegme et autre	12

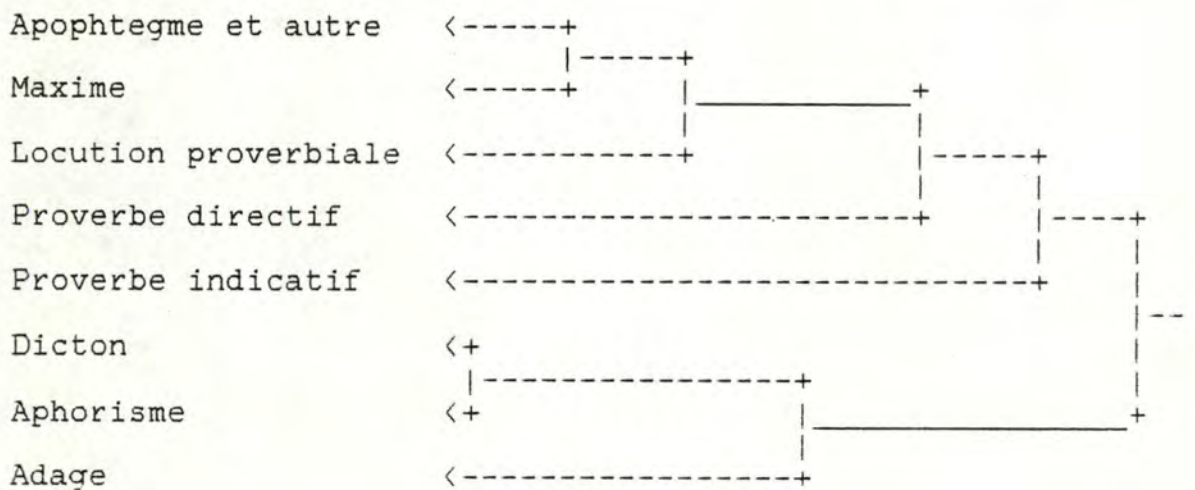
Inertie expliquée : 46.72% + 29.23% = 75.95% de l'inertie totale

Table de contingence en valeurs absolues

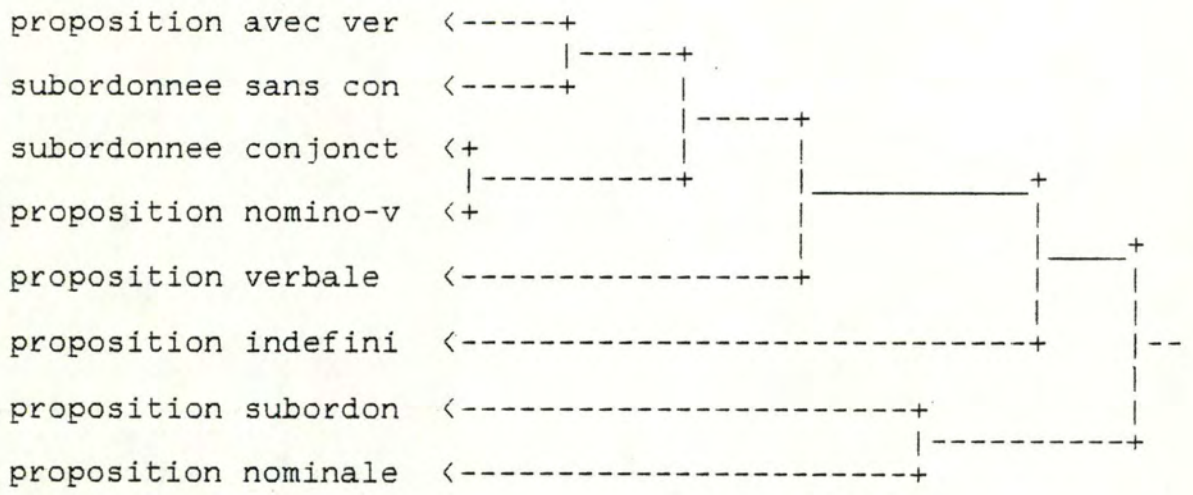
Proverbe indicatif	481	275	384	400	11	90	192	1336	3169
Proverbe directif	21	17	17	10	4	4	9	33	115
Maxime	8	8	10	10	1	2	6	16	61
Aphorisme	323	125	223	315	14	23	86	613	1722
Locution proverbiale	9	3	8	14	1	0	6	16	57
Dicton	2	0	1	2	0	0	0	4	9
Adage	12	1	1	3	0	0	0	9	26
Apophtegme et autre	5	5	5	12	0	2	6	14	49
Total colonnes	861	434	649	766	31	121	305	2041	5208

proposition nominale
proposition nomino-verbale
proposition verbale
proposition subordonnee
subordonnee sans conjonction
subordonnee conjonctionnelle
proposition avec verbe aux.
proposition indefinie
Total lignes

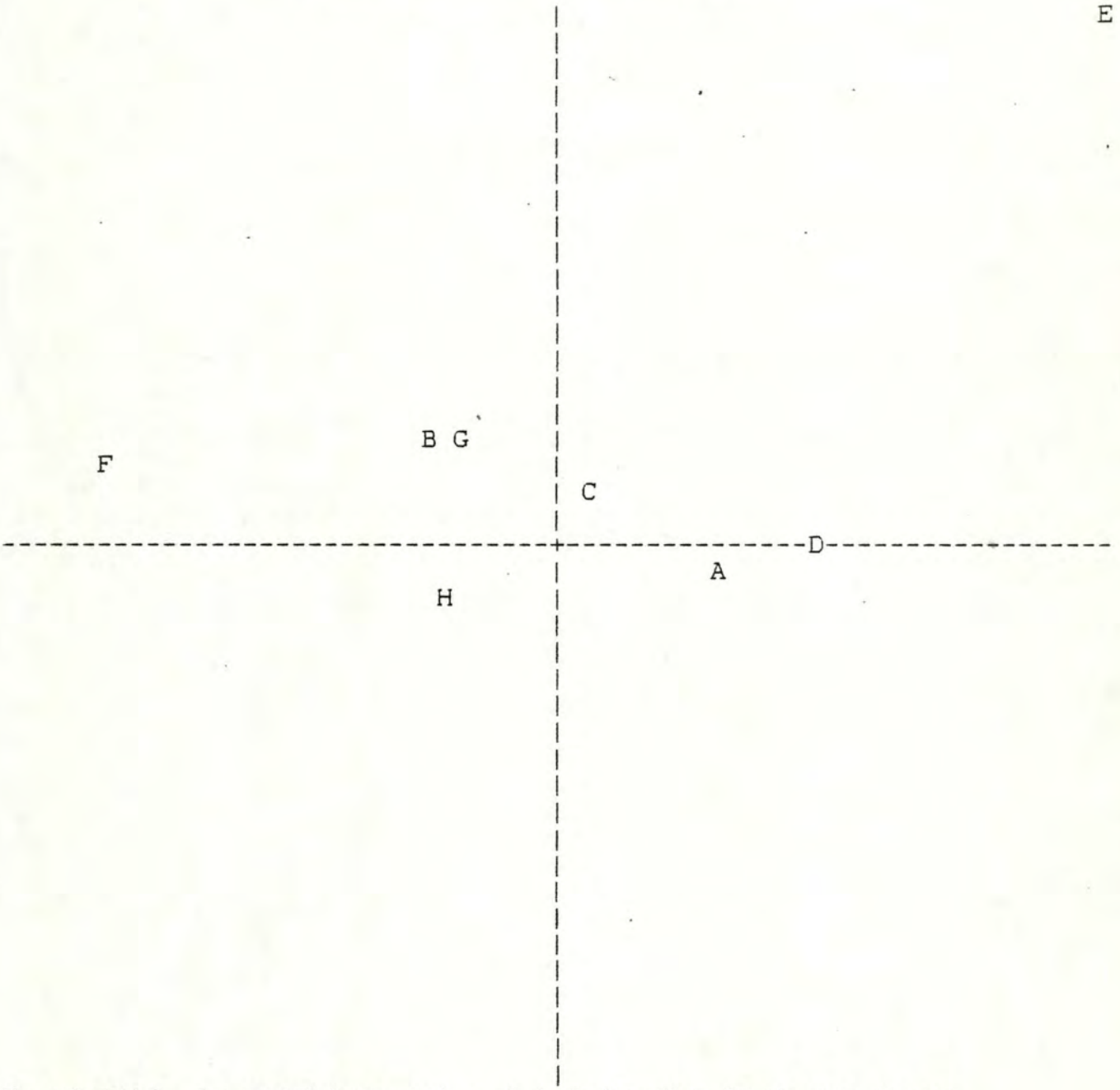
Hiérarchisation sur base de la variance



Hiérarchisation sur base de la variance



Analyse des correspondances

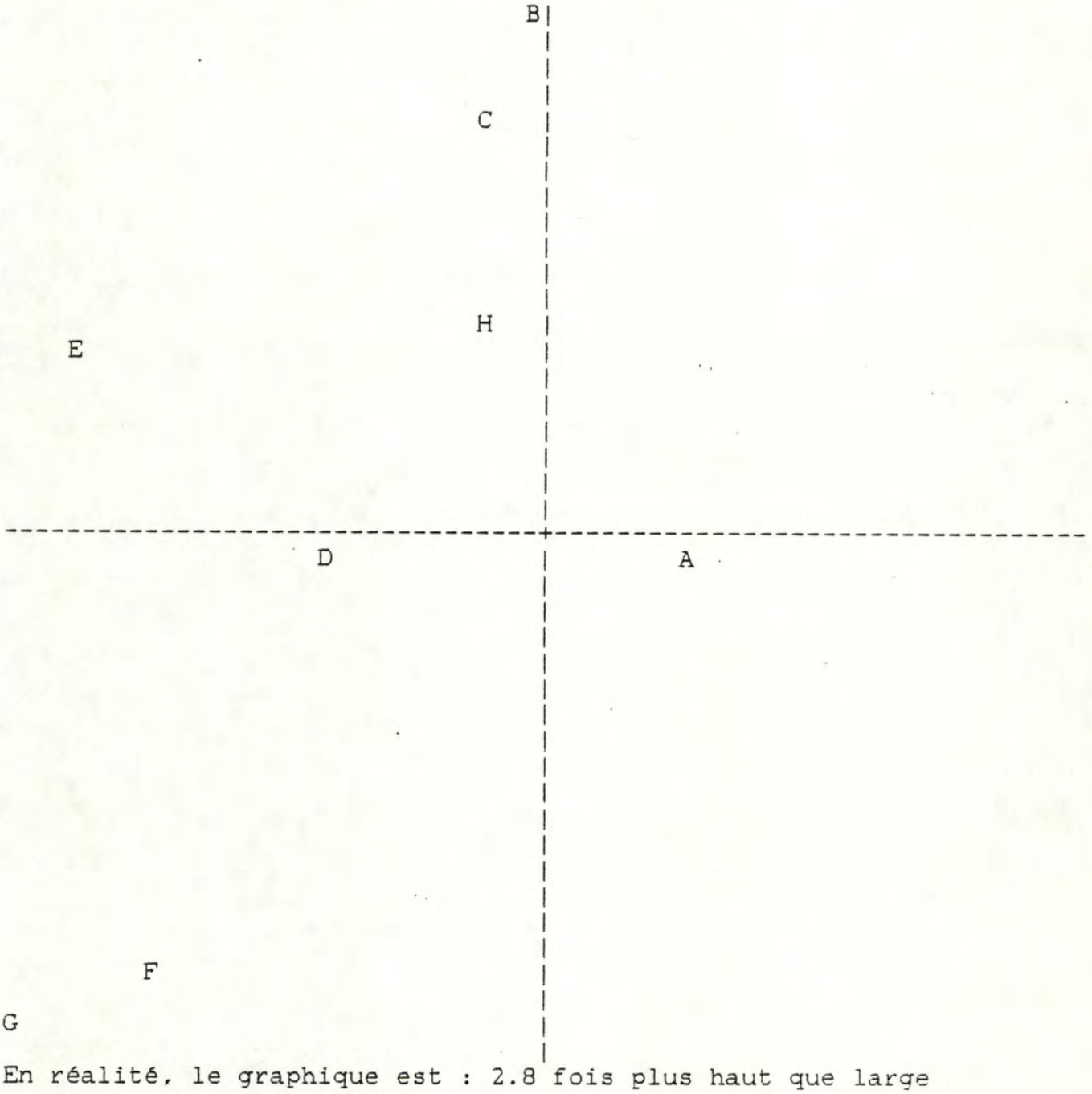


En réalité, le graphique est : 4.1 fois plus haut que large

A:proposition nominale	861	B:proposition nomino-verbale	434
C:proposition verbale	649	D:proposition subordonnee	766
E:subordonnee sans conjonction	31	F:subordonnee conjonctionnelle	121
G:proposition avec verbe aux.	305	H:proposition indefinie	2041

Inertie expliquée : 54.03% + 29.14% = 83.17% de l'inertie totale

Analyse des correspondances



En réalité, le graphique est : 2.8 fois plus haut que large

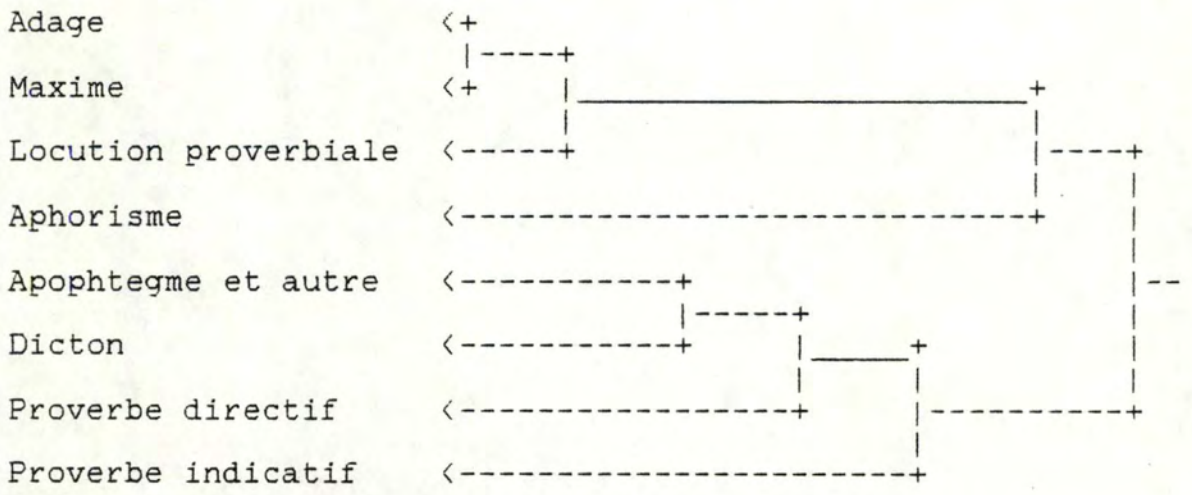
A:Proverbe indicatif	3169	B:Proverbe directif	115
C:Maxime	61	D:Aphorisme	1722
E:Locution proverbiale	57	F:Dicton	9
G:Adage	26	H:Apophtegme et autre	49

Inertie expliquée : 54.03% + 29.14% = 83.17% de l'inertie totale

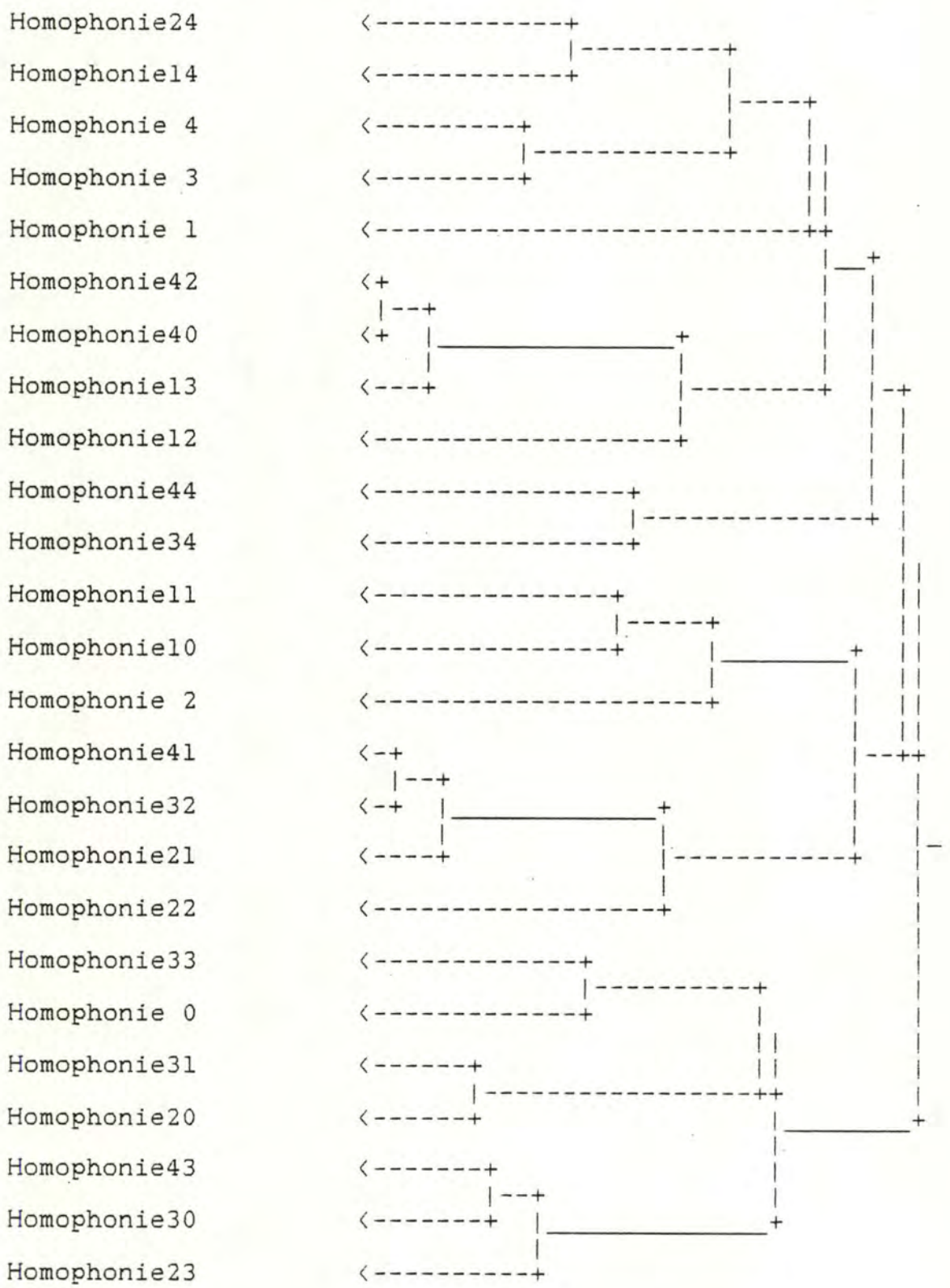
Table de contingence en valeurs absolues

	Proverbe indicatif	Proverbe directif	Maxime	Aphorisme	Locution proverbiale	Dicton	Adage	Apophtegme et autre	Total lignes
Homophonie 0	416	12	14	289	13	0	7	5	756
Homophonie 1	334	17	3	144	1	0	2	7	508
Homophonie 2	321	11	7	144	4	0	4	3	494
Homophonie 3	108	5	1	39	3	1	1	0	158
Homophonie 4	36	2	1	10	1	0	0	0	50
Homophonie10	98	2	4	46	2	0	0	1	153
Homophonie11	183	5	6	81	4	0	3	3	285
Homophonie12	257	7	0	96	4	1	0	2	367
Homophonie13	187	4	2	66	1	0	0	0	260
Homophonie14	83	4	1	41	1	2	1	2	135
Homophonie20	61	1	2	59	2	0	1	3	129
Homophonie21	103	2	0	59	0	0	0	1	165
Homophonie22	165	4	3	89	0	0	0	4	265
Homophonie23	114	5	1	79	1	0	1	0	201
Homophonie24	66	2	1	35	1	1	0	0	106
Homophonie30	10	0	0	13	0	0	1	0	24
Homophonie31	10	0	1	14	1	0	0	1	27
Homophonie32	48	0	0	32	0	0	0	1	81
Homophonie33	48	2	1	37	3	0	1	0	92
Homophonie34	54	1	0	33	2	2	0	3	95
Homophonie40	6	0	0	3	0	0	0	0	9
Homophonie41	4	0	0	3	0	0	0	0	7
Homophonie42	8	0	0	3	0	0	0	0	11
Homophonie43	15	1	0	18	0	0	0	0	34
Homophonie44	22	0	1	17	0	2	0	1	43
Total colonnes	2757	87	49	1450	44	9	22	37	4455

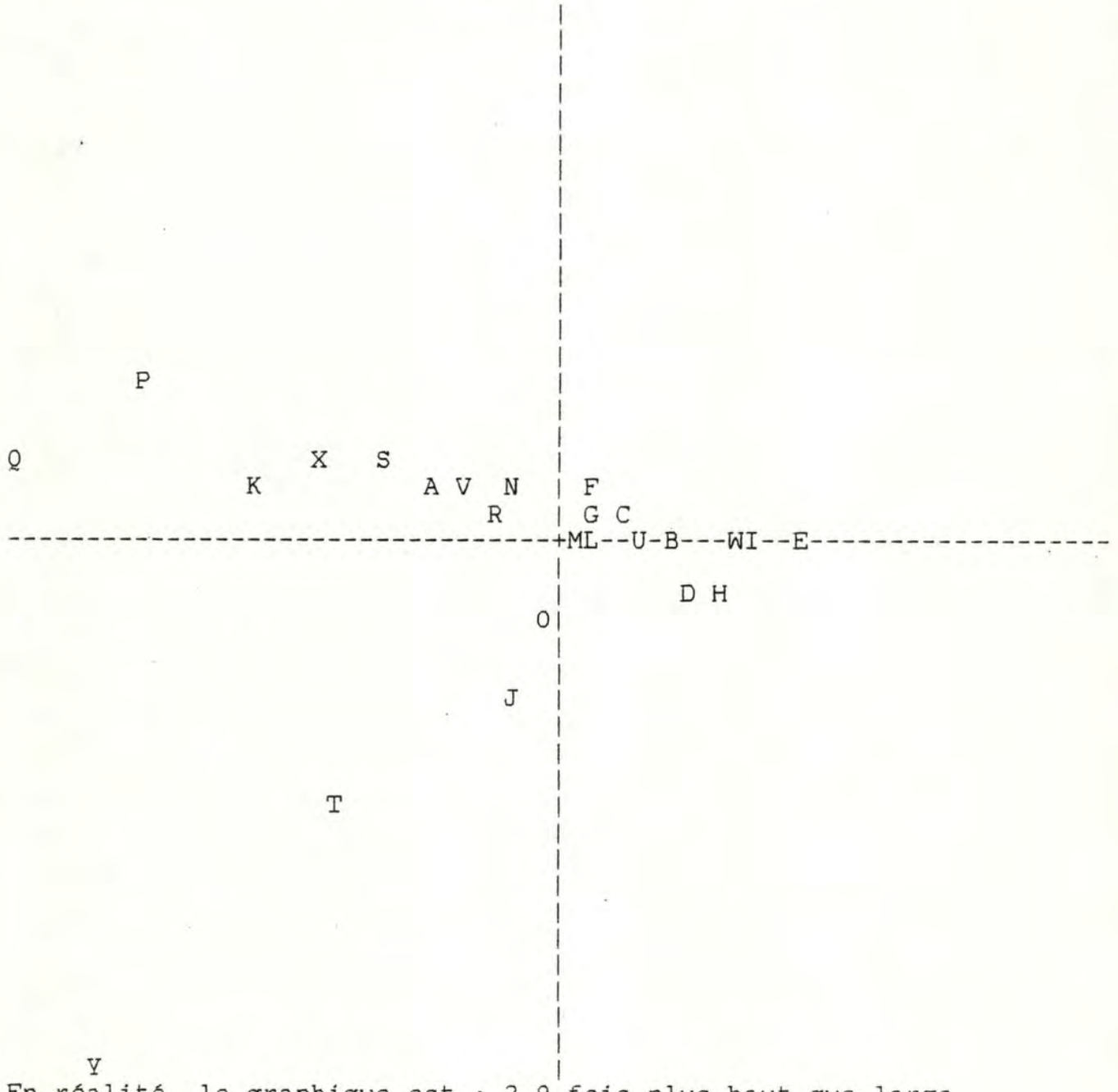
Hiérarchisation sur base de la variance



Hiérarchisation sur base de la variance



Analyse des correspondances

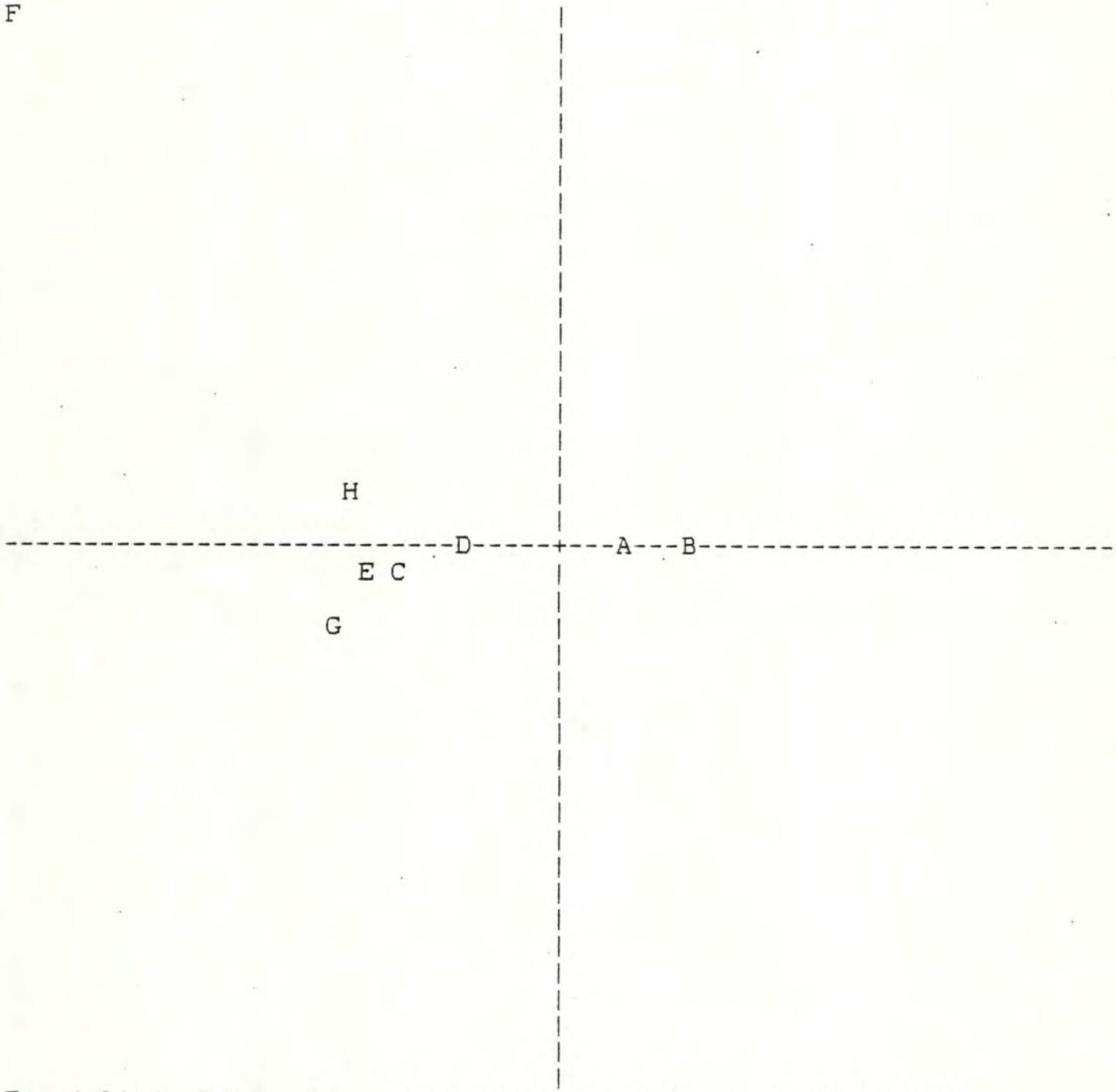


En réalité, le graphique est : 2.9 fois plus haut que large

A:Homophonie 0	756	B:Homophonie 1	508	C:Homophonie 2	494
D:Homophonie 3	158	E:Homophonie 4	50	F:Homophonie10	153
G:Homophonie11	285	H:Homophonie12	367	I:Homophonie13	260
J:Homophonie14	135	K:Homophonie20	129	L:Homophonie21	165
M:Homophonie22	265	N:Homophonie23	201	O:Homophonie24	106
P:Homophonie30	24	Q:Homophonie31	27	R:Homophonie32	81
S:Homophonie33	92	T:Homophonie34	95	U:Homophonie40	9
V:Homophonie41	7	W:Homophonie42	11	X:Homophonie43	34
Y:Homophonie44	43				

Inertie expliquée : 45.37% + 38.91% = 84.28% de l'inertie totale

Analyse des correspondances



En réalité, le graphique est : 5.6 fois plus haut que large

A:Proverbe indicatif	2757	B:Proverbe directif	87
C:Maxime	49	D:Aphorisme	1450
E:Locution proverbiale	44	F:Dicton	9
G:Adage	22	H:Apophtegme et autre	37

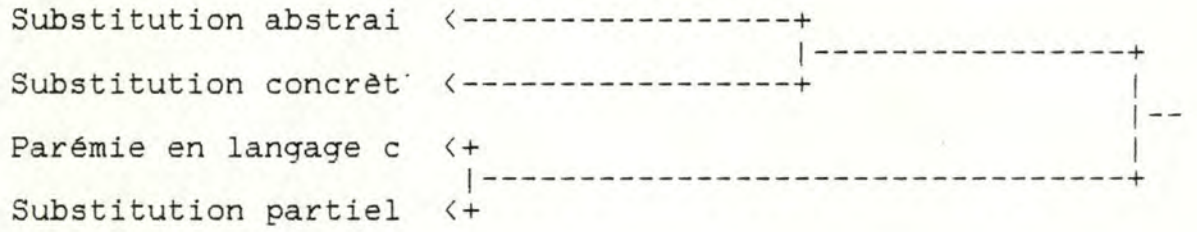
Inertie expliquée : 45.37% + 38.91% = 84.28% de l'inertie totale

Table de contingence en valeurs absolues

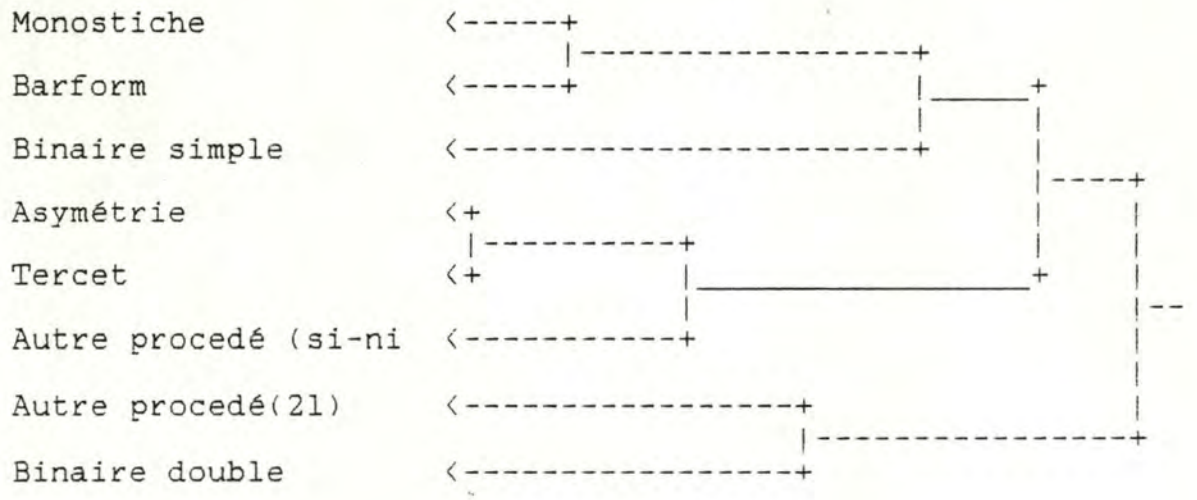
Binaire simple	8	73	82	58	221
Binaire double	160	887	272	131	1450
Tercet	1	7	5	3	16
Barform	2	4	3	0	9
Autre procedé (si-niko)	129	1198	586	242	2155
Monostiche	22	25	17	9	73
Autre procedé(21)	44	210	56	22	332
Asymétrie	11	106	50	32	199
Total colonnes	377	2510	1071	497	4455

Substitution concrète
Substitution abstraite
Substitution partielle
Parémie en langage clair
Total lignes

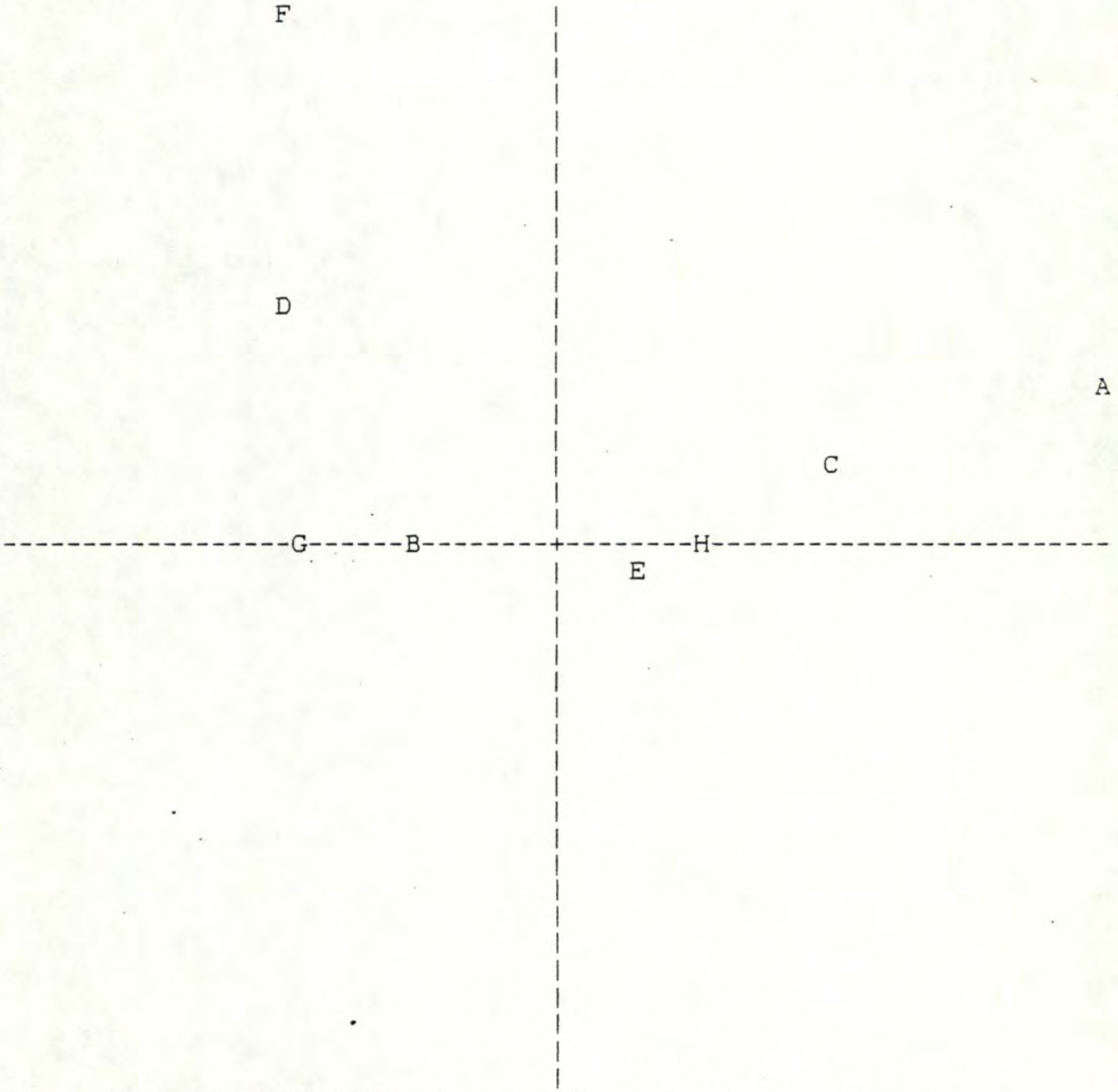
Hiérarchisation sur base de la variance



Hiérarchisation sur base de la variance



Analyse des correspondances

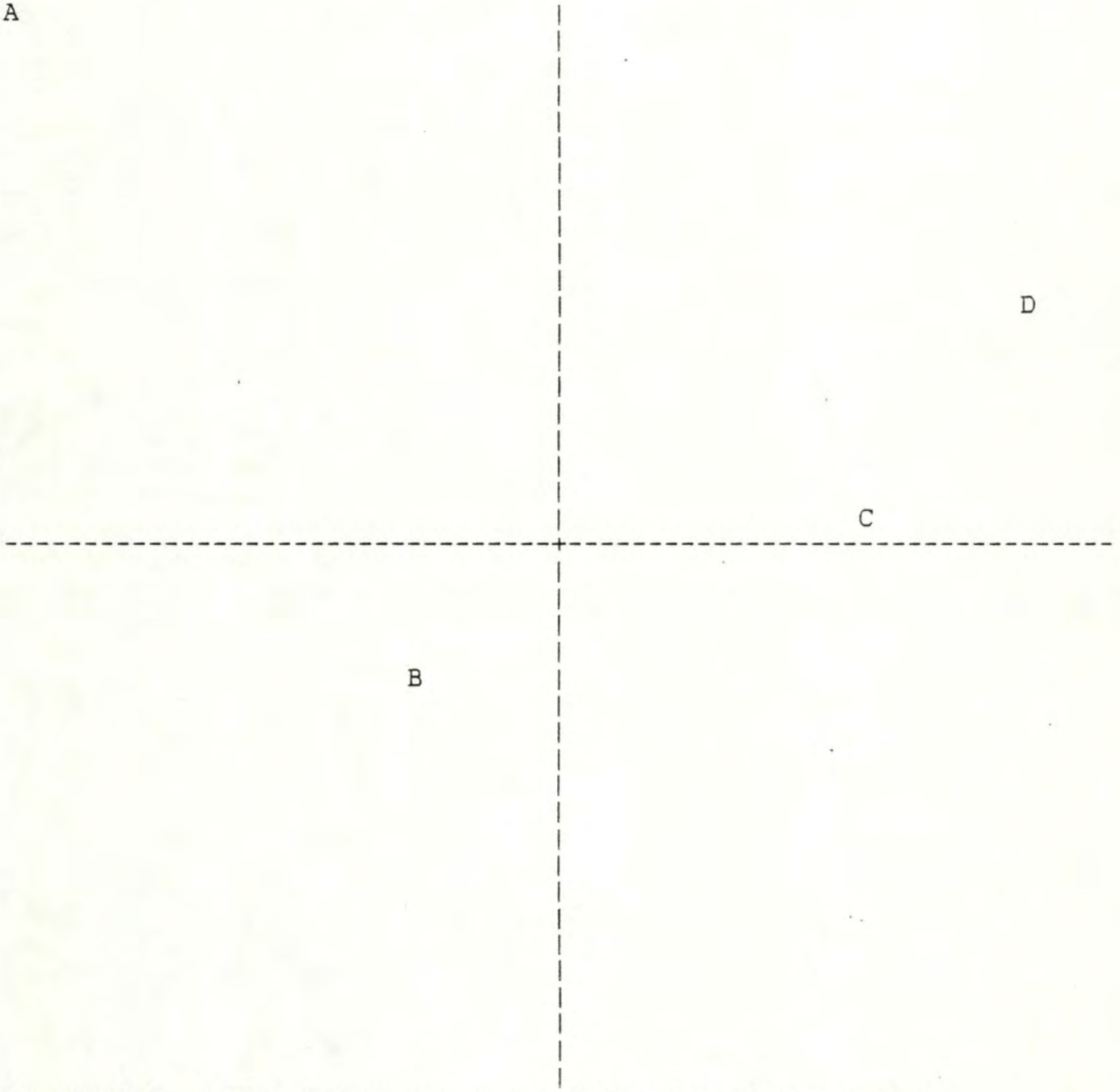


En réalité, le graphique est : 3.6 fois plus haut que large

A:Binaire simple	221	B:Binaire double	1450
C:Tercet	16	D:Barform	9
E:Autre procedé (si-niko)	2155	F:Monostiche	73
G:Autre procedé(21)	332	H:Asymétrie	199

Inertie expliquée : 71.27% + 24.50% = 95.77% de l'inertie totale

Analyse des correspondances



En réalité, le graphique est : 2.3 fois plus haut que large

A:Substitution concrète	377	B:Substitution abstraite	2510
C:Substitution partielle	1071	D:Parémie en langage clair	497

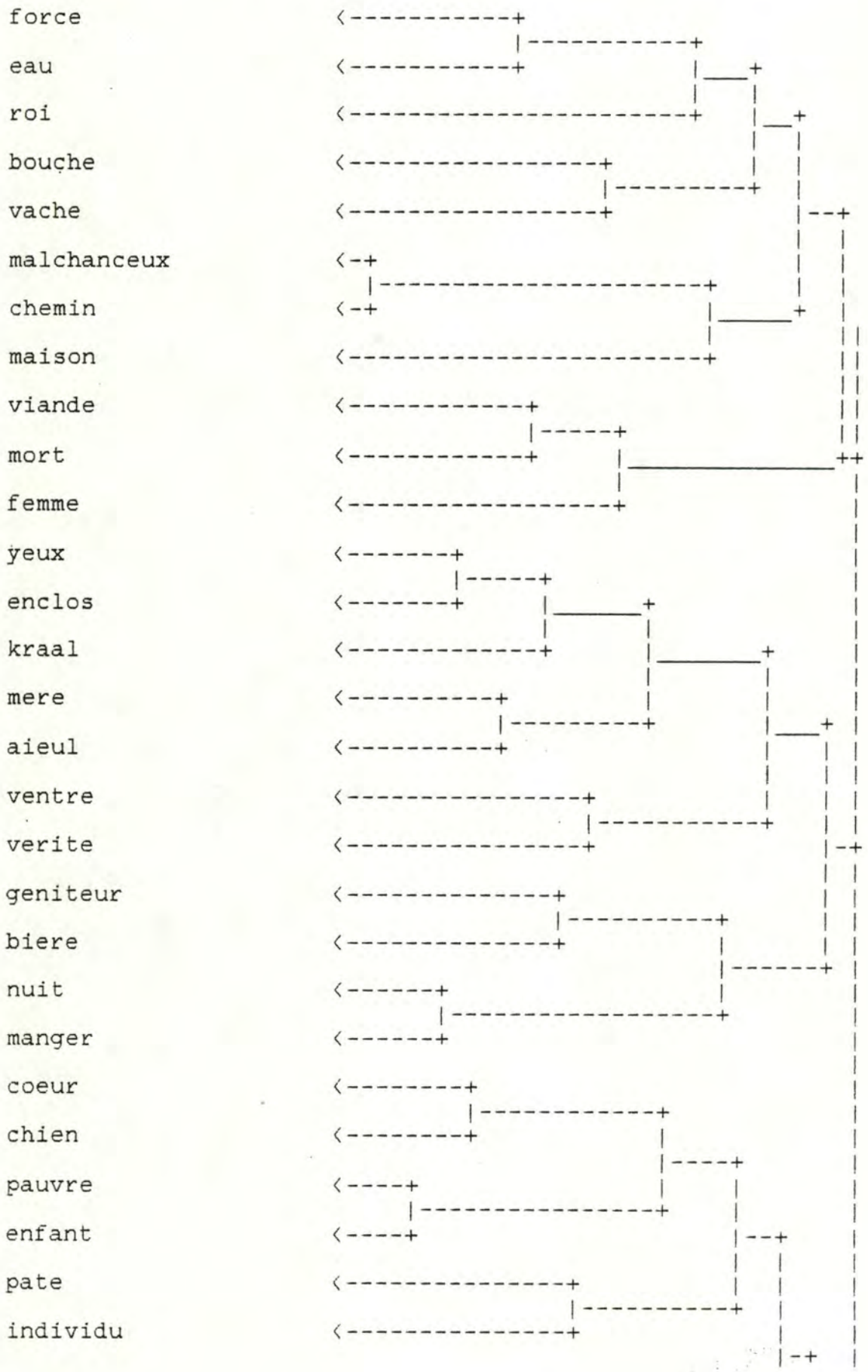
Inertie expliquée : 71.27% + 24.50% = 95.77% de l'inertie totale

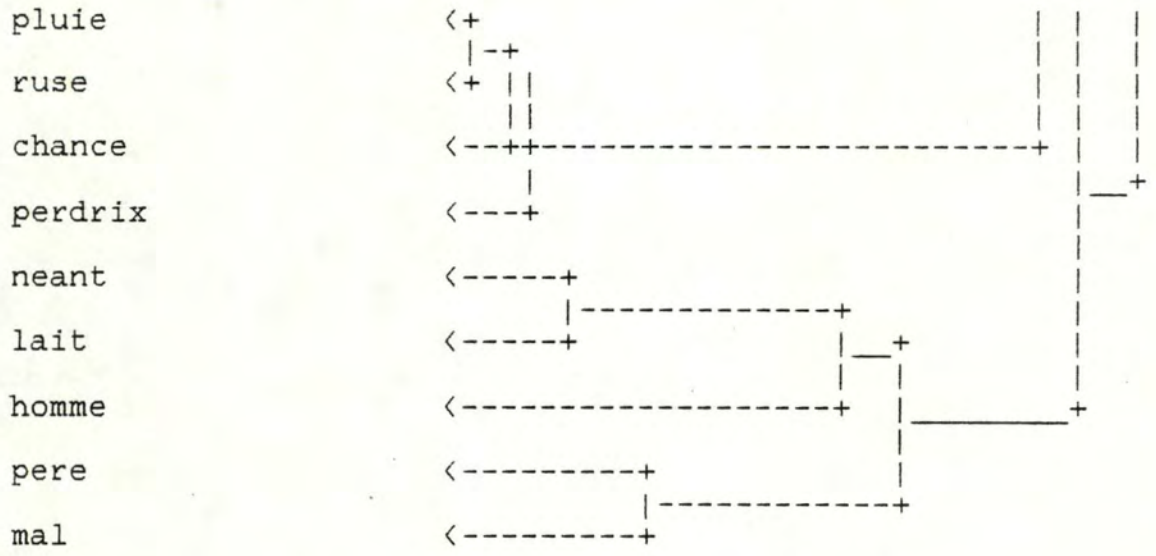
Table de contingence en valeurs absolues

homme	6	40	186	8	0	240
enfant	4	48	214	2	0	268
chance	0	0	8	0	0	8
individu	4	15	83	2	0	104
vache	6	9	53	3	0	71
chien	1	9	29	1	0	40
verite	12	19	46	0	0	77
roi	9	2	53	0	0	64
ventre	16	34	79	0	0	129
enclos	4	14	35	2	0	55
femme	3	23	77	20	0	123
pauvre	0	5	22	0	0	27
eau	10	3	35	0	0	48
mal	0	13	51	4	0	68
pate	1	7	50	0	0	58
biere	0	12	16	2	0	30
coeur	1	10	34	0	0	45
manger	0	13	27	0	0	40
ruse	0	0	5	0	0	5
chemin	6	3	12	4	0	25
lait	1	4	23	2	0	30
geniteur	0	10	14	0	0	24
maison	14	18	47	6	0	85
bouche	6	7	31	1	0	45
aieul	2	7	16	0	0	25
mere	12	32	79	3	0	126
pere	0	3	19	1	0	23
mort	0	1	16	3	0	20
kraal	2	6	21	1	0	30
force	7	2	27	1	0	37
pluie	0	0	5	0	0	5
viande	0	4	16	3	0	23
yeux	1	7	14	1	0	23
perdrix	0	0	28	0	0	28
nuit	1	11	26	0	0	38
malchanceux	0	0	0	0	0	0
neant	2	6	32	2	0	42
Total colonnes	131	397	1529	72	0	2129

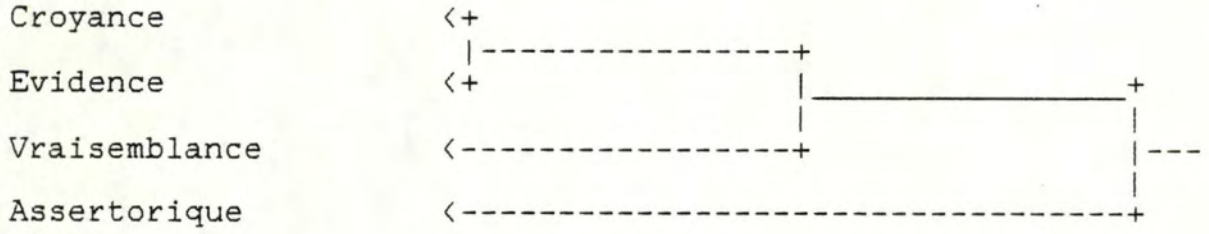
Evidence
Vraisemblance
Assertorique
Croyance
Affirmation gratuite
Total lignes

Hiérarchisation sur base de la variance

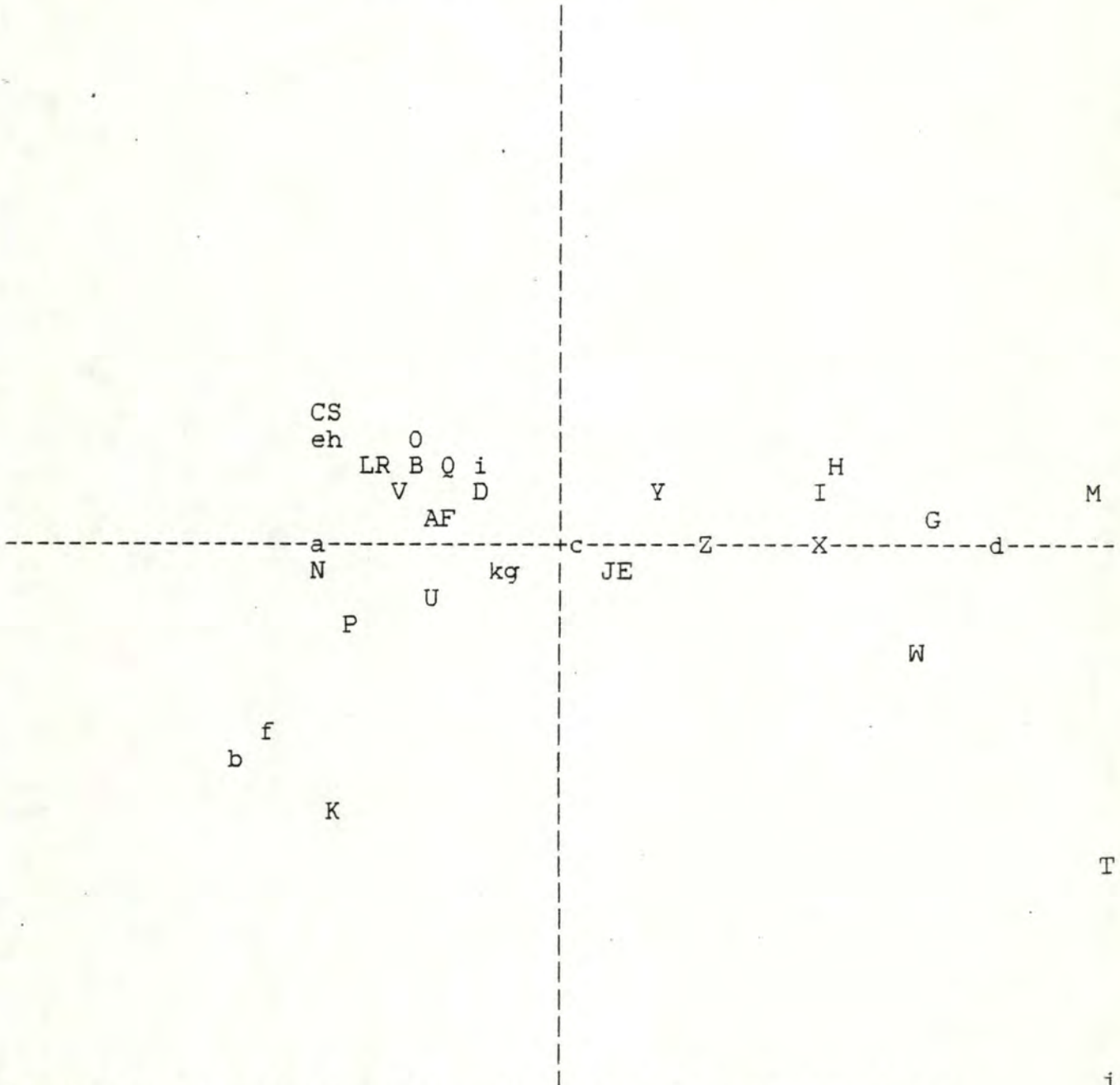




Hiérarchisation sur base de la variance



Analyse des correspondances

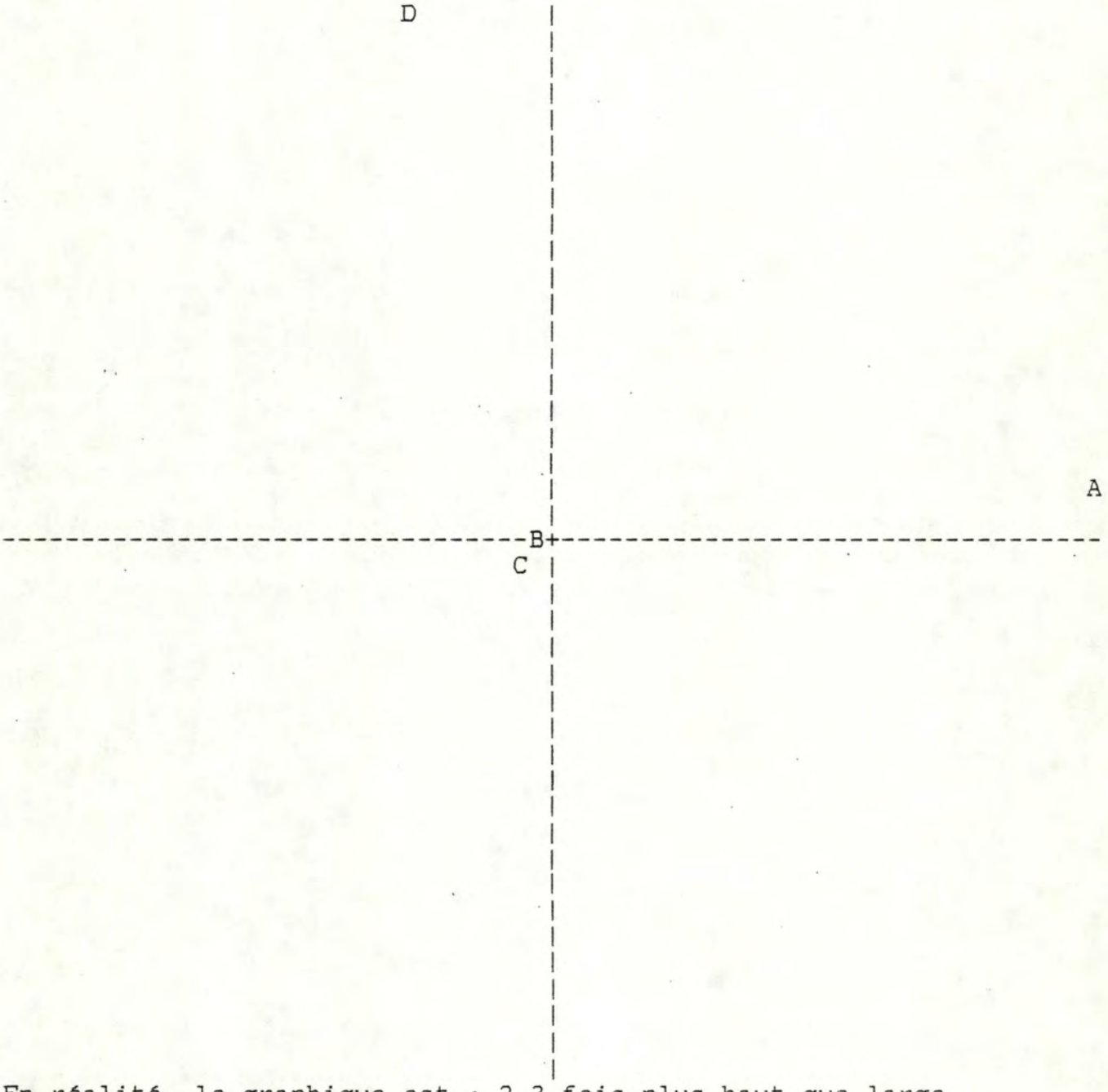


En réalité, le graphique est : 3.9 fois plus haut que large

A:homme	240	B:enfant	268	C:chance	8	D:individu	104
E:vache	71	F:chien	40	G:verite	77	H:roi	64
I:ventre	129	J:enclos	55	K:femme	123	L:pauvre	27
M:eau	48	N:mal	68	O:pate	58	P:biere	30
Q:coeur	45	R:manger	40	S:ruse	5	T:chemin	25
U:lait	30	V:geniteur	24	W:maison	85	X:bouche	45
Y:aieul	25	Z:mere	126	a:pere	23	b:mort	20
c:kraal	30	d:force	37	e:pluie	5	f:viande	23
g:yeux	23	h:perdrix	28	i:nuit	38	j:malchanceux	0
k:neant	42						

Inertie expliquée : 39.94% + 36.30% = 76.23% de l'inertie totale

Analyse des correspondances



En réalité, le graphique est : 2.3 fois plus haut que large

A:Evidence	131	B:Vraisemblance	397
C:Assertorique	1529	D:Croyance	72

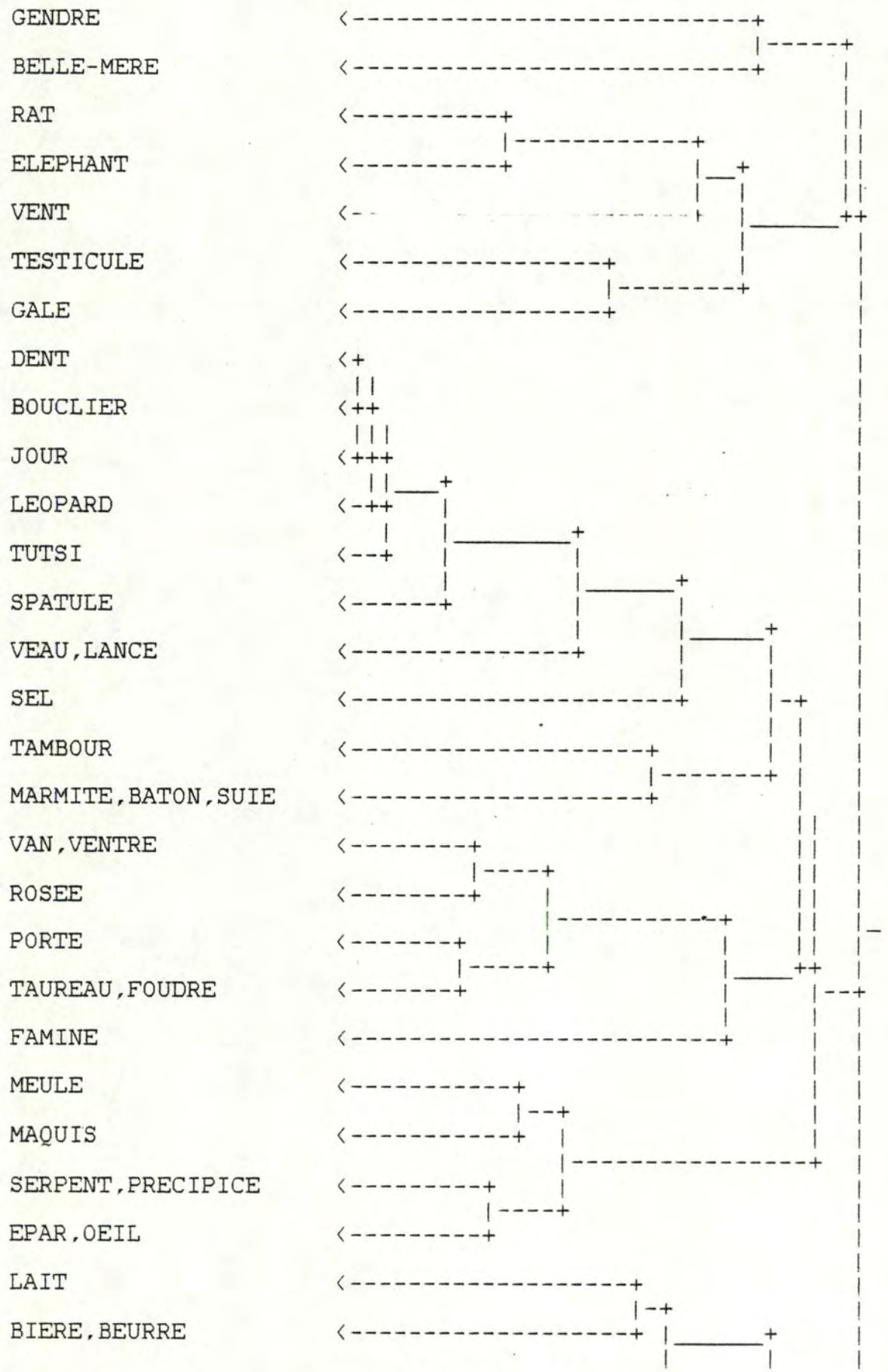
Inertie expliquée : 39.94% + 36.30% = 76.23% de l'inertie totale

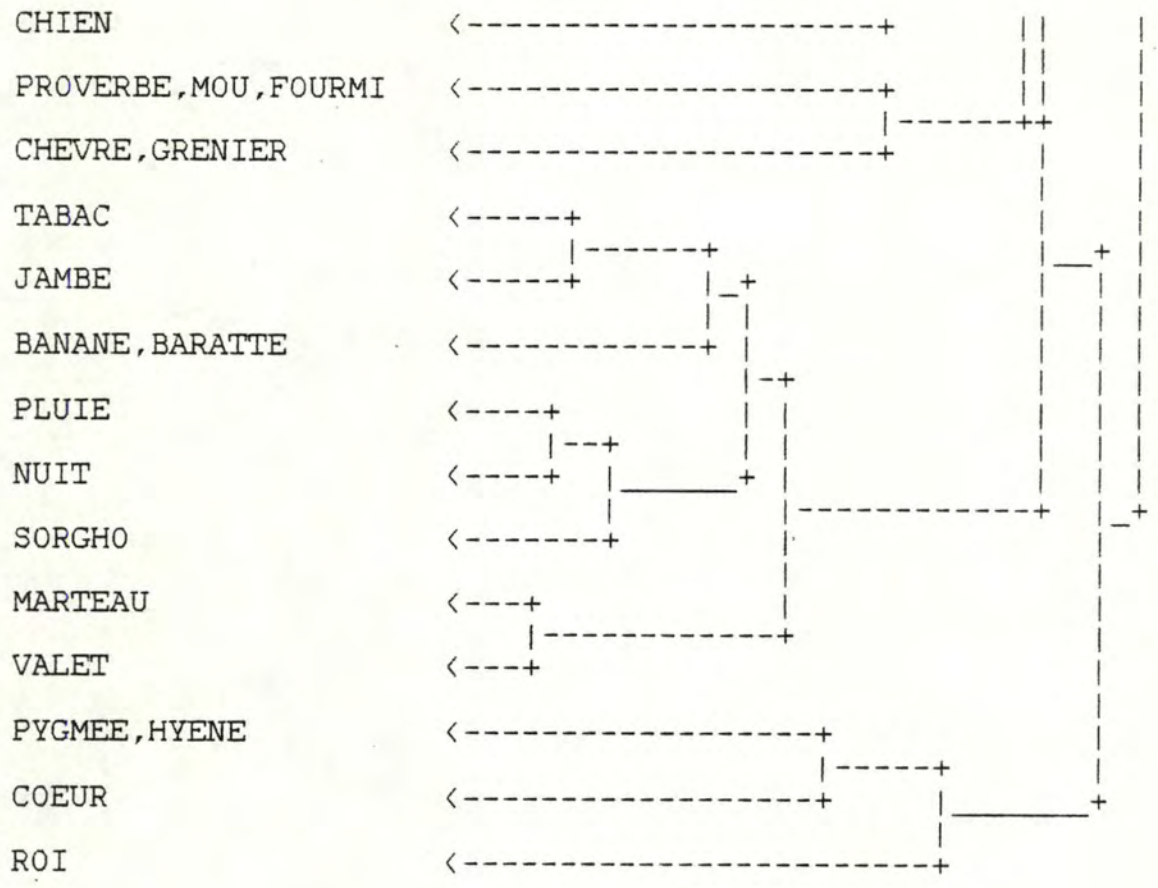
Table de contingence en valeurs absolues

BANANE,BARATTE	0	0	6	0	0	6
MARMITE,BATON,SUIE	2	6	9	0	0	17
BELLE-MERE	0	0	4	4	0	8
BIERE,BEURRE	0	1	10	0	0	11
BOUCLIER	0	0	0	0	0	0
CHEVRE,GRENIER	2	0	16	0	0	18
CHIEN	1	2	16	0	0	19
COEUR	1	1	4	0	0	6
DENT	0	0	0	0	0	0
ELEPHANT	4	0	1	0	0	5
EPAR,OEIL	0	1	0	0	0	1
FAMINE	0	5	3	0	0	8
TAUREAU,FOUDRE	0	1	1	0	0	2
PROVERBE,MOU,FOURMI	2	1	17	0	0	20
GALE	2	0	1	0	0	3
GENDRE	0	0	0	2	0	2
VALET	0	0	1	0	0	1
PYGMEE,HYENE	3	2	9	0	0	14
JAMBE	0	0	2	0	0	2
JOUR	0	0	0	0	0	0
LAIT	0	1	5	0	0	6
VEAU,LANCE	2	4	5	1	0	12
LEOPARD	0	0	0	0	0	0
MAQUIS	0	3	0	0	0	3
MARTEAU	0	0	1	0	0	1
MEULE	0	2	0	0	0	2
NUIT	0	0	3	0	0	3
PLUIE	0	0	3	0	0	3
PORTE	0	1	1	0	0	2
SERPENT,PRECIPICE	0	1	0	0	0	1
RAT	4	0	1	0	0	5
ROI	3	1	11	0	0	15
ROSEE	0	2	2	0	0	4
SEL	1	1	1	0	0	3
SORGHO	0	0	3	0	0	3
SPATULE	0	0	0	0	0	0
TABAC	0	0	2	0	0	2
TAMBOUR	0	1	2	0	0	3
TESTICULE	1	0	1	0	0	2
TUTSI	0	0	0	0	0	0
VAN,VENTRE	0	2	2	0	0	4
VENT	2	0	0	0	0	2
Total colonnes	30	39	143	7	0	219

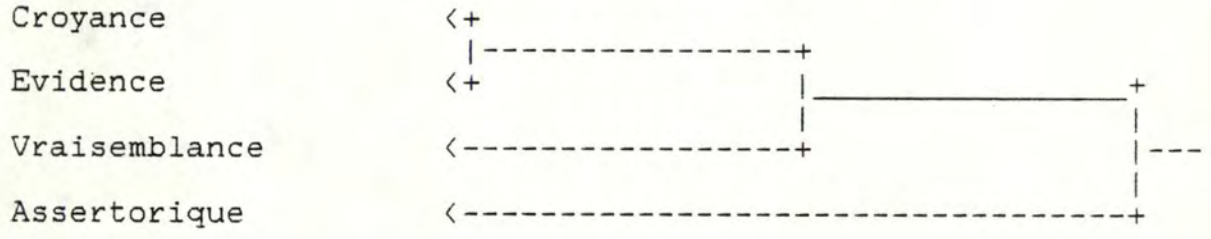
Evidence
Vraisemblance
Assertorique
Croyance
Affirmation gratuite
Total lignes

Hiérarchisation sur base de la variance

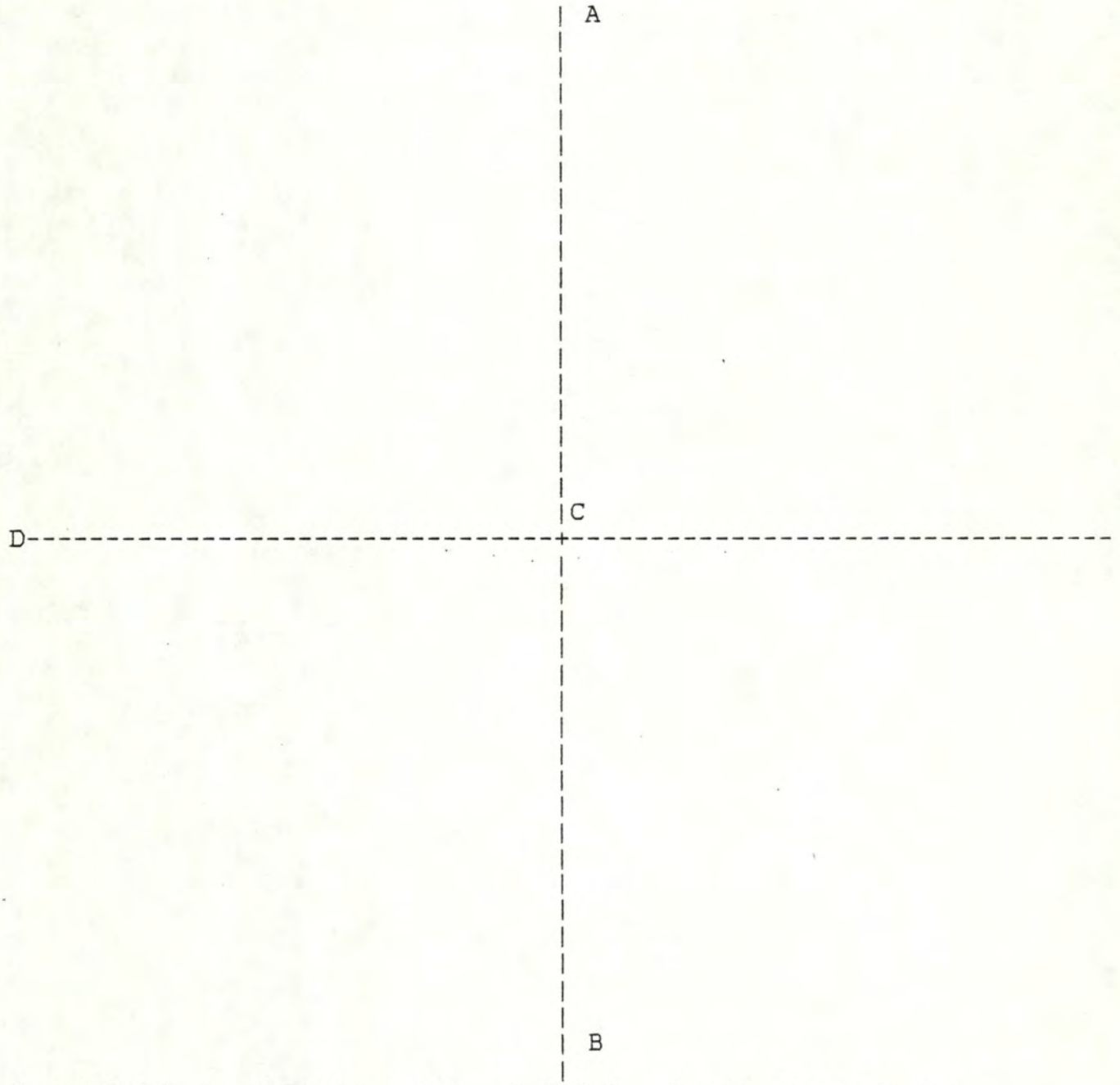




Hiérarchisation sur base de la variance



Analyse des correspondances



En réalité, le graphique est : 1.8 fois plus haut que large

A:Evidence	30	B:Vraisemblance	39
C:Assertorique	143	D:Croyance	7

Inertie expliquée : 45.77% + 29.87% = 75.64% de l'inertie totale

1 EXEMPLE-TYPE

L'exemple suivant est tiré de [Ber81, ch. VIII]

Le tableau des effectifs (que nous noterons K) est très peu pratique pour comparer le profil des lignes ou des colonnes entre elles. Pour cela, on préfère construire le tableau des fréquences relatives (noté F).

Tableau 3-4 : Fréquences relatives

Valeur d'attention	2.8	3.2	1.6	3.5	2.3	3.5	16.9
Esthétique	1.9	1.9	2.7	2.1	1.3	1.9	11.7
Contenu informatif	0.4	1.1	0.3	0.6	1.8	0.5	4.7
Interêt de la promesse	1.1	1.4	1.1	1.4	0.7	2.0	7.6
Implication	1.6	1.0	1.8	1.5	0.9	1.7	8.5
Crédibilité	1.2	2.3	1.2	0.6	0.4	1.6	7.2
Directionnalité image	1.7	2.5	1.4	1.8	0.7	2.6	10.6
Directionnalité titre	1.7	1.1	1.8	0.8	1.1	2.0	8.4
Compréhension apparente	1.2	0.9	1.5	1.1	1.7	1.8	8.1
Incitation à réagir	1.4	1.5	1.5	0.1	0.4	1.4	6.3
Compréhension réelle	0.9	2.7	1.3	1.8	0.7	2.6	10.0
Total colonnes	15.8	19.5	16.2	15.2	11.7	21.5	100.0
	n°1 A	n°2 N	n°3 N	n°4 O	n°5 N	n°6 C	Total E S lignes

On remarque que la comparaison de lignes ou de colonnes est plus aisée qu'avec un tableau de fréqueneces relatives. L'idéal cependant pour comparer les colonnes ou les lignes entre elles serait de les diviser par leur totaux respectifs. On obtient ainsi le tableau 3-5 des profils-lignes et le tableau 3-6 des profils-colonnes.

Tableau 3-5 : Profils-lignes

Valeur d'attention	16.4	18.7	9.6	21.0	13.5	20.8	100.0
Esthétique	16.0	16.4	22.8	17.5	10.8	16.4	100.0
Contenu informatif	9.3	23.4	6.5	12.1	38.3	10.3	100.0
Interêt de la promesse	13.8	19.0	14.4	18.4	8.6	25.9	100.0
Implication	18.7	11.9	20.7	18.1	10.4	20.2	100.0
Crédibilité	16.4	32.1	16.4	8.5	4.8	21.8	100.0
Directionnalité image	16.0	23.0	13.6	16.9	6.2	24.3	100.0
Directionnalité titre	19.9	12.6	21.5	9.4	13.1	23.6	100.0
Compréhension apparente	15.1	10.8	18.4	13.0	20.5	22.2	100.0
Incitation à réagir	22.4	23.8	24.5	2.1	5.6	21.7	100.0
Compréhension réelle	8.8	26.8	13.2	17.5	7.5	26.3	100.0
Total colonnes	172.8	218.4	181.4	154.6	139.3	233.4	1100.0
	n°1 A	n°2 N	n°3 N	n°4 O	n°5 N	n°6 C	Total E S lignes

Tableau 3-6 : profils-colonnes

Valeur d'attention	17.5	16.2	10.0	23.3	19.4	16.3	102.7
Esthétique	11.9	9.9	16.5	13.5	10.8	9.0	71.6
Contenu informatif	2.8	5.6	1.9	3.7	15.3	2.2	31.6
Interêt de la promesse	6.7	7.4	6.8	9.2	5.6	9.2	44.8
Implication	10.0	5.2	10.8	10.1	7.5	7.9	51.4
Crédibilité	7.5	11.9	7.3	4.0	3.0	7.3	41.0
Directionnalité image	10.8	12.6	8.9	11.8	5.6	12.0	61.7
Directionnalité titre	10.6	5.4	11.1	5.2	9.3	9.2	50.7
Compréhension apparente	7.8	4.5	9.2	6.9	14.2	8.4	50.9
Incitation à réagir	8.9	7.6	9.5	0.9	3.0	6.3	36.1
Compréhension réelle	5.6	13.7	8.1	11.5	6.3	12.2	57.4
Total colonnes	100.0	100.0	100.0	100.0	100.0	100.0	600.0
	n°1 A	n°2 N	n°3 N	n°4 O	n°5 N	n°6 C	Total E S lignes

Chaque opinion est caractérisée par un nombre de ménagères qui ont cette opinion pour une annonce particulière. Ainsi deux opinions seront très proches l'une de l'autre si elles ont plus ou moins même suffrage de ménagères pour les annonces respectives. Calculer la proximité des opinions i et j revient donc à calculer la DISTANCE entre les lignes i et j du tableau 3-5

Semblablement, chaque annonce est caractérisée par le nombre de ménagères qui possèdent telle et telle opinion sur cette annonce. Deux annonces seront fort "semblables" si plus ou moins le même nombre de ménagères ont les mêmes opinions sur cette annonce. Donc calculer la ressemblance entre deux annonces du point de vue des opinions que les ménagères en ont, revient à calculer la DISTANCE entre deux colonnes du tableau 3-6 .

Mathématiquement chaque point-ligne (opinion) est un point dans l'espace à 6 dimensions des annonces. Inversément, chaque annonce est un point dans l'espace à 11 dimensions des opinions. L'ensemble des opinions constitue donc un nuage de points dans l'ensemble à six dimensions des annonces (et vice-versa).

En outre si la table de contingence représente un échantillon d'une population nous pouvons interpréter les profils-lignes et colonnes en termes probabilistes. Cette possibilité ne sera pas explorée ici, car les parémies du Burundi se rapprochent beaucoup plus d'une population que d'un échantillon[1].

(2) Cfr [rbf309] et [rbb132] (3) Pour démonstration, [rbf309-310]

2 DÉFINITION DE LA DISTANCE.

Une distance[1] -ou métrique- D entre deux individus A et B d'une même population P , que l'on note D(A,B), est un nombre non strictement positif .

Soient A,B,C trois individus d'une même population P, on dispose des trois axiomes suivants :

1. $D(A,B) = D(B,A)$ (Symétrie)
2. $D(A,B) = 0 \iff A = B$
3. $D(A,B) + D(B,C) = D(A,C)$ (Inégalité triangulaire)
4. Si $D(A,B) \leq \max(D(A,B), D(B,C))$ alors cette distance sera appelée <u>ultra-métrique</u>[2]

Si la distance euclidienne nous est la plus familière, il existe une infinité de types de distances répondant à la définition ci-dessus,

notamment :

1. La variance : il s'agit en fait de l'écart par rapport à la moyenne.
2. La distance du X^2 : elle possède des propriétés intéressantes dont nous reparlerons.

Soient deux vecteurs[1] X et Y appartenant à un même espace,

$$D^2(X(i),X(j)) = \sum_{j=1}^{n} \frac{nbind\ (\ K(i,j)/\ K(i) - K(j,k)/\ K(k)^2}{K(j)}$$

avec nbind : nombre d'individus de la population
K : table des fréquences absolues
K(i) : total de la ième ligne de K
K(j) : total de la jème colonne de K
K(k) : total de la kème ligne de K
n : le nombre de colonnes de K

(2) cfr [Leb79] p 388

3 FORMULATION MATRICIELLE DE LA DISTANCE

Calculer la distance entre deux vecteurs[1] X et Y peut se faire d'une manière matricielle en appliquant la formule

$$D(x,y) = \sqrt{(x-y)' \, Q \, (x-y)}$$

Q est la matrice des distances ; c'est elle qui définit la métrique utilisée.

4 INERTIE D'UN NUAGE

L'inertie d'un nuage N par rapport à Y appartenant à R, que l'on note I(y) est définie par

$$\sum_{i=1}^{n} p(i) \, D^2(x(i),Y)$$

5 CALCUL DES AXES PRINCIPAUX

Mathématiquement, le nième axe principal est un vecteur propre du produit matriciel de K'D K M (K est la matrice des fréquences absolues et V = K' D K la matrice d'inertie du nuage par rapport à son centre de gravité).
L'inertie totale du nuage est la somme des valeurs propres de VM. L'inertie expliquée par le jème axe principal est la part de la jème plus grande valeur propre dans le total des valeurs propres.

(1) Ligne ou colonne d'une matrice

6 PRÉ-TESTS X^2.

Le test X^2 permet de faire une telle pré-étude et de se prononcer sur la dépendance entre les lignes et les colonnes d'une table de contingence de deux critères. Nous nous contenterons ici d'en rappeler les grands principes. Nous renvoyons le lecteur à [rbp322] pour un plus ample développement.

Soit A, une table de contingence à n lignes et p colonnes entre deux critères quelconques établie sur une population P. On peut construire la matrice[1] des fréquences théoriques de A que nous noterons T.

Il s'agit d'une matrice qui possède les caractéristiques suivantes :

1. Elle possède le même effectif que A
2. Tous ses profils-lignes sont identiques
3. Tous ses profils-colonnes sont identiques

Ainsi par exemple la matrice des fréquences théoriques de l'exemple du chapitre III est la suivante.

Table de contigence entre les classes des critères 1 et 2

Valeur d'attention	63	72	37	81	52	80	385
Esthétique	43	44	61	47	29	44	268
Contenu informatif	10	25	7	13	41	11	107
Interêt de la promesse	24	33	25	32	15	45	174
Implication	36	23	40	35	20	39	193
Crédibilité	27	53	27	14	8	36	165
Directionnalité image	39	56	33	41	15	59	243
Directionnalité titre	38	24	41	18	25	45	191
compréhension apparente	28	20	34	24	38	41	185
Incitation à réagir	32	34	35	3	8	31	143
comprehension réelle	20	61	30	40	17	60	228
Total des colonnes	360	445	370	348	268	491	2282
	n°1	n°2	n°3	n°4	n°5	n°6	Total des
	A	N	N	O	N	C	E	S	lignes

Cette matrice présente la particularité d'avoir tous ses points-colonnes ainsi que tous ses points-lignes confondus en un même point. Ce serait le cas limite où toutes les ménagères auraient exactement la même opinion sur toutes les annonces. Cette matrice est dite théorique parce que ce serait celle qui serait obtenue si toutes les classes d'un critère étaient identiques sur base de l'autre critère. Elle est la plus favorable à l'hypothèse d'indépendance des lignes et des colonnes.

(1) Dans ce chapitre, le mot matrice sera un synonyme de table de contingence

Il est possible en considérant la table de contingence obtenue de calculer l'écart de la réalité par rapport à la théorie. Il suffit pour cela de calculer la somme des écarts entre chaque élement de la matrice des effectifs avec l'élément correspondant au carré (pour avoir une quantité positive). En outre, on pondère cet écart en le divisant par la proportion théorique. De cette manière, l' écart provenant de classes de faible effectif théorique aura, dans le total D^2, un poids inférieur à l'écart des classes d'effectif théorique plus important. Nous appelerons la quantité ainsi obtenue D^2.

$$\sum_{i=1}^{n} \sum_{j=1}^{p} \frac{(K(i,j) - T(i,j))^2}{T(i,j)}$$

On peut prouver que la distribution statistique de ce D^2 suit une loi de probabilité X^2 à (n - 1) x (p - 1) degrés de liberté.

$$\lim_{n \to \infty} pr[D^2 <= a] = F_{X^2_{(n-1) \times (p-1)}}(a)$$

On teste donc l'homogénéité de la matrice K. L'épreuve consiste à rejeter l'hypothèse d'homogéité si

$$D^2 > Q_{X^2_{(n-1) \times (p-1)}}(1 - a)$$

Il est possible de calculer la probabilité de dépassement ,c-à-d la probabilité que D^2 soit plus grand qu'une valeur donnée dans l'hypothèse d'indépendance des lignes et des colonnes.

$$PD = 1 - F_{X^2_{(n-1) \times (p-1)}}(D^2)$$

Il existe un moyen rapide de trouver la valeur de D^2 lorsqu'on a déja calculé les valeurs propres de la matrice K.

$$D^2 = N \times \text{trace}(K) \qquad (2)$$

avec N, le nombre d'individus sous-épreuves
trace(K), la somme des valeurs propres de la matrice K

C'est cette dernière formule qui sera appliquée. La vérification des valeurs obtenues pourra ainsi être comparée avec les résultats fournis par les programmes en C (cfr I.1) ; en outre c'est aussi un excellent moyen de vérifier le bon fonctionnement du module des valeurs propres

Les résultats obtenus sont les suivants :

Croisement des critères 2 et 3 :	4496.26	4496.65
3 et 4 :	373.25	341.09
2 et 4 :	235.38	237.04

Remarque

Deux critères ne sont pas pris en ligne de compte. Le premier parce qu'un classement thématique n'avait pas été envisagé à l'époque. Le cinquième parce qu'il ne porte que sur 2000 parémies.

La première colonne montre le résultat du programme en langage C qui applique la formule traditionnelle ; la deuxième le résultat du programme en pascal qui applique la formule (2). Le léger écart entre les résultats peut s'expliquer par les erreurs d'arrondis dues au calcul des valeurs propres.

Ces valeurs de X^2 sont énormes et montrent bien à quel point les classes engendrées par la classification originale sont peu homogènes.